Ára: 30 Ft

μ mikro számítógép magazin

ABLAK — gitt nélkül
(17. oldal)

1989/11.

Sci-L

IBM PC/XT-, AT-, 386-KOMPATIBILIS PROFESSZIONÁLIS SZEMÉLYI SZÁMÍTÓGÉPEK

AJÁNLATUNKBAN ÚJ

P—132—386 alapkonfiguráció

ÁR: 349 000,— Ft+ÁFA
80386; 20/25 MHz
4 MB RAM
80 MB winchester (28 ms)
Monokróm monitor, 12"
1,2 MB floppy
Billentyűzet (101 gombos)
Torony kivitel

P—16/CA—286 AT alapkonfiguráció

ÁR: 220 000,— Ft+ÁFA
80286; 10/20 MHz
2 MB RAM
40 MB winchester (28 ms)+csatoló
Monokróm monitor, 12"
1,2 MB floppy
Billentyűzet (101 gombos)

P—16/CA-286 AT alapkonfiguráció

ÁR: 153 000,— Ft+ÁFA
80286; 6/10 MHz
1 MB RAM
20 MB winchester (ST225)+csatoló
Monokróm monitor, 12"
1,2 MB floppy
Billentyűzet

P—16/PC alapkonfiguráció

ÁR: 80 000,— Ft+ÁFA
8088; 4,77/10 MHz
640 kB RAM
Billentyűzet (84 gombos)
Hercules/színes csatoló
Monokróm monitor, 12"
360 kB floppy
Real-time óra
Soros-párhuzamos interfész

P—16/CX XT alapkonfiguráció

ÁR: 105 000,— Ft+ÁFA
8088; 4,77/10 MHz
640 kB RAM
20 MB winchester (ST225)
Monokróm monitor, 12"
360 kB floppy
Billentyűzet (84 gombos)

Garancia nélküli reklámáraink:

• TRS aszinkron terminál+kábel	9 000,— Ft+ÁFA
• TRS mátrixnyomtató+kábel (132 kar/sor)	14 500,— Ft+ÁFA
• FX—100 mátrixnyomtató+kábel (132 kar/sor)	49 900,— Ft+ÁFA
• RANK XEROX 4045 lézernyomtató	290 000,— Ft+ÁFA
• UNIBOARD billentyűzet (101 gombos)	4 900 ,— Ft+ÁFA

Bővítési lehetőségeinkről kérjék árjegyzékünket!

SZÁMÍTÁSTECHNIKAI INFORMATIKAI FEJLESZTŐ LEÁNYVÁLLALAT

1011 Budapest, Iskola u. 10.
Telex: 22 4599
Telefax: 35 39 15

Sci-L

Telefon: 1-154-065
1-350-180/180, 181, 182, 184

7. ÉVFOLYAM
1989/11. SZÁM

mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság vezetője:
Kovács Győző

A szerkesztőség munkatársai:
Bakos Tamás
(programozástechnika)
Broczkó Péter
(hírek)
Kovács Győző
(levelezés)
Nagy Imre
(tanuljuk együtt)
Petróczy Judit
(könyvek)
Pinke György
(Enterprise)
Soltészné Vizi Zsuzsa
(tervezőszerkesztő)
Szebenszki Sándor
Tamásné Lakó Erika
Terebessy Ákosné
Varga János
(olvasószerkesztő)

Címképünk:
Velekey József Lajos munkája

Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 115-4250

Levélcím:
1371 Budapest Pf. 433

Kiadja:
MTESZ Neumann János Számítógéptudományi Társaság
1054 Budapest, Báthori u. 16.

Levélcím:
1368 Budapest 5. Pf. 240

Telefon: 132-9349

Felelős kiadó:
Tóth Istvánné ügyvezető főtitkárhelyettes

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlap-előfizetési és Lapellátási Irodáján
(1900 Budapest XIII., Lehel u. 10/A)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta.
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, Pf. 149
és a Magyar Média
1932 Budapest, Pf. 279
88—1135

89 - 2681 - Révai Nyomda, Budapest
Felelős vezető:
Horváth Józsefné dr.

INDEX: 25 629
ISSN 0236-6088

TARTALOM

2 Egy konferencia ürügyén
9 Feladatok — megoldások
36 Útmutató
38 Az AmigaBasic utasításkészlete
41 Amire jó a PIA
44 Programtermék — Egy újabb angolnyelv-oktató program
45 Adok — veszek — cserélek

TANULJUK EGYÜTT! 3

3 A Pascal rejtelmei
5 Feladatmegoldás számítógéppel

CSIPEGETŐ 11

11 Zak McKRACKEN and the alien mindbenders III.
11 Magyarországon is shareware
12 Pillanatfelvétel a piacról
14 Biztos betöltés
14 TOP-lista

PROGRAMOZÁSTECHNIKA 15

15 BASIC-bővítések Commodore 16-ra
17 Ablakozó
20 Programozási fogások és melléfogások

ENTERPRISE 21

21 Csatlakozzunk, de mivel?
23 A gőgös spectrumos esete az Enterprise-zal
26 A LORIGRAPH rajzolóprogram hasznosítása II.
26 Mi a manó?

PÉCÉZZÜNK! 27

27 RAM-rezidens programok
31 Tud-e a DOS magyarul?
33 Soros vonal a PC-n

SAKK 43

43 Az értékelőfüggvény hatékonyságának elemzése

KÖNYVEK — HÍREK — ÉRDEKESSÉGEK 46

AZ OLVASÓ ÍRJA 48

„Szeretném életemet arra szentelni, hogy a világ rólunk alkotott véleménye a helyes irányba változzék, és hogy egyáltalán legyen rólunk vélemény a világban."
(Latinovits Zoltán: Jó estét kívánok)

Egy konferencia ürügyén

Hivatalos üzleti ügyeimet sikerült úgy intéznem, hogy két tárgyalás között részt vehettem a manchesteri második nemzetközi számítástechnika-történeti konferencián, ahol két, egymást követő szekcióban — az amerikaiban és az európaiban — meghívott előadók ismertették a számítástechnika ős- és hőskorának nevezetesebb fejezeteit.

Élvezettel ültem végig a két napot, már csak azért is, mert 1990 októberében nálunk rendezzük meg az *IFIP History Commission* első munkakonferenciáját, és így alkalmam volt néhány szakembert személyesen is meghívnom. Arról van szó ugyanis, hogy Amerikában már évtizedek óta foglalkoznak a számítástechnika történetével, és gyűjtik a szakma pótolhatatlan emlékeit. (Ilyenkor nem tudok nem gondolni a szétszedett M—3-ra!) Amerikában van a Babbage Institute is, amelynek igazgatója a manchesteri konferencia záróakkordjaként emelkedett hangú beszédben méltatta a számítástechnika történeti kutatásának jelentőségét.

A korábbi konferencián az előadók többsége elsősorban a kezdeti amerikai fejlesztésekkel és eredményekkel foglalkozott, azt az érzést keltve a hallgatóban, mintha a korai számítástechnika csak az Egyesült Államokban létezett volna. Ebben persze van is némi igazság. A mostani konferencia egy kicsit más volt, a rendezők megpróbáltak egészséges egyensúlyt találni az amerikai és az európai témák között. (Ne felejtsük el, hogy elsősorban a számítástechnika-történet korai szakaszáról — úgy a hatvanas évek közepéig — volt szó.) Néhány előadó azonban nem „röstellte" hangoztatni, hogy az akkori európai fejlesztés nem volt nagyon gazdag eredeti gondolatokban, inkább csak követte az amerikai mintát. Japánról meg egyáltalán nem beszéltek, egyöntetűen azt állítva, hogy a számítástechnika őstörténetét illetően ez az ország nem vehető számításba. (Nekünk legyen mondva, én sem bánnám, ha Japánhoz hasonlóan nem volna számítástechnikai múltunk, hanem hozzájuk hasonló jövőnk.)

A kétnapos ülés nemcsak érdekes, de számomra nagyon tanulságos is volt, és tulajdonképpen megerősítette egy korábban már elmondott és le is írt véleményemet, hogy az elmúlt évek hibás társadalmi és gazdasági politikája, valamint ennek következménye — azaz elszigeteltségünk a világ fejlettebbik részétől — több kárt okozott nekünk, de a többi szocialista országnak is, mint a ma annyit emlegetett kölcsönfelvétel, Bős—Nagymaros vagy bármilyen más hibás gazdasági döntés.

Volt egy nagyon érdekes előadás a francia számítástechnika korai szakaszáról, kb. 1965-ig bezárólag. Ebben az időszakban Franciaország elsősorban a háború okozta anyagi és erkölcsi károk felszámolásával volt elfoglalva, de akkori politikusai arról sem feledkeztek meg, hogy a lyukak foltozgatása nem elég, ugyanis ha utol akarják érni a háborúban kevésbé sérült Anglia vagy az USA gazdaságát, akkor nagyon nagy mértékben, szinte az erejüket meghaladó módon kell támogatniuk a műszaki fejlesztést és a kutatást. Egy kivérzett országnak is „lépést kell tartani" a többi iparilag fejlett vagy esetleg fejlettebb országgal, ez volt a ki nem mondott jelszó, és mindig akadt olyan politikus De Gaulle-tól Mitterand-ig, aki messzire látott, és elég erős is volt ahhoz, hogy a visszahúzó és csak a rövid távú érdekekre koncentráló erőket legyőzze. Az eredmény a mai francia számítástechnika szintjén pontosan lemérhető. Így azután van számítógép-hálózatuk, de van európai hírű telefonhálózatuk is. Miután volt elfogadható iskolaszámítógép-koncepciójuk, és nem „dőltek be" az olcsó, de a célnak egyáltalán nem megfelelő, tehát használhatatlan külföldi házi számítógépes ajánlatoknak, hanem rendszerben gondolkozva kifejlesztették a feltételeknek legjobban megfelelő iskolai rendszereket, elmondhatják, hogy náluk az iskola-számítástechnika megoldottnak tekinthető. Így volt ez sok más területen is, ezért nem tűnt hihetetlennek az előadó kijelentése, hogy — szerinte — a háború óta nem volt olyan vezető politikus, aki ne a műszaki szakemberek véleményének megfelelően döntött volna ezekben, az ország boldogulását rendkívüli mértékben befolyásoló kérdésekben.

Ebben az évben „ünnepeljük" az első hazai számítógép, az M—3 üzembe állításának 30. évfordulóját, ami — úgy látszik — csupán néhányunkat érdekel. Tulajdonképpen én is némi röstelkedéssel foglalkozom az évfordulóval, egy kicsit úgy tűnhet ugyanis, mintha azt a csoportot szeretném ünnepeltetni, amelynek magam is tagja voltam. Pedig csak arról van szó, és ez a konferencián többször is elhangzott, hogy az igazán jelentős eredmények eléréséhez a műszaki múlt ismerete nagyon is szükséges. Talán ennek bizonyítéka, hogy az iparilag fejlett országok egyre több egyetemén és főiskoláján tanítják már az informatika történetét a számítástechnika szakos hallgatóknak. Ezen a konferencián is feltűnt egy ifjú előadó, aki a doktori disszertációjának egy részletét adta elő, sikeresen.

Hallgatva az előadásokat, nem lehetett nem észrevenni, hogy abban az időben, amikor az M—3-at csináltuk, nem is voltunk mi annyira lemaradva az európai, de még a világ számítástechnikája mögött sem. Alig néhány év különbség volt „csak" az EDVAC, az IAS (Neumann—Goldstine-féle gép) és az M—3 között, nem is beszélve a velünk majdnem egy időben szárnyat bontogató más nyugati országok számítástechnikájáról. Talán Ausztria járt jóval előttünk, hiszen Heinz Zemanek már akkor elkészült a Mailufferrel, a világ első teljesen tranzisztorizált számítógépével, vagy az angolok az EDSAC-kal, amit végeredményben az EDVAC európai változatának lehet tekinteni, ennek ellenére az angolok nagyon büszkék rá, hiszen mégis csak ez volt az első, Angliában épült és működő számítógép. Még az ezt követő években sem voltak rosszak a sanszaink, hiszen egy viszonylag hosszabb szünet után elkészült az EMG 803-as, a KFKI-ban a TPA sorozat, de az eredmények közé kell sorolnunk a soha be nem fejezett EDLA-koncepciót is, amelylyel ismét csak sikerült valamit lefaragnunk az elmaradásunkból. Én azt hiszem, nem járok messze az igazságtól, ha az ezt követő években egyre növekvő elmaradást (nemcsak a miénket, de a többi szocialista országét is) a politika számlájára írom, hiszen ekkor kezdődött a hidegháború, elszigetelődésünk a világ fejlettebbik részétől, és vált intézményesítetté a bizalmatlanság, szűnt meg a technológiai együttműködés, nem vásárolhattunk fejlett számítástechnikai eszközöket, saját erőből pedig képtelenek voltunk mindent kifejleszteni.

Üldögélve a konferencián, hallgatva az előadókat, óhatatlanul előrekalandoztam az időben: azt próbáltam megfogalmazni, mit is kellene most csinálnunk. Sajnos, ma is a politikusokon múlik minden, hiszen nekik kell először visszaszerezniük az elvesztett bizalmat, és hazánkat mint egy nyitott demokratikus országot elfogadtatniuk a nagyvilággal. A magyar számítástechnika híre-neve nem rossz, ma már szinte közhelynek számít szoftverfejlesztőink termékeire hivatkozni, de vannak hardverfejlesztési eredményeink is, amelyeket nem is egyszer csak egy külföldi fejlesztő intézetben lehet megvalósítani, hiszen sem nálunk, sem a többi szocialista országban nem állnak rendelkezésre azok a technológiai körülmények, amelyek a fejlesztéshez feltétlenül szükségesek.

Azt hiszem, hogy tudnak rólunk a világban, talán nincs is rossz véleményük Magyarországról, sőt, mintha a vélemény — leszámítva egy-egy szegletét a világnak — egyre javulna velünk kapcsolatban. Most úgy érzem, nagyon sok múlik rajtunk, hogy a kivívott tekintélyünket kamatoztatni tudjuk, mert a mérkőzés második fordulója már nem a politikusoké, hanem a miénk: a műszakiaké, az orvosoké, az íróké, az újságíróké, általában a szakembereké, és persze nem utolsósorban a tanítóké, tanároké. Talán nem tűnik nagyképűségnek, ha megkockáztatom, hogy rajtunk már nem fog múlni a dolog!

Kovács Győző

A PASCAL REJTELMEI

15. Grafikus segédfájlok alkalmazása

Az előző részben már szó volt arról, hogy a grafikai feladatok megkönnyítésére rendelkezésre áll a GRAPH.P és a GRAPH.BIN fájl. Ezek közül a GRAPH.P azoknak a grafikus eljárásoknak és függvényeknek a hívását végzi, amelyek gépi kódú formában a GRAPH.BIN fájlban találhatók. Mivel összesen 41 eljárás és függvény van — bár ezek egy részét a Turbo-fordító is tartalmazza —, nem törekedhetünk a teljességre. Az ismertetésre nem kerülő — mindössze néhány — dolog „kibogozását" az olvasóra bízzuk.

A grafikus segédfájlok lehetőségeit szemlélteti a *táblázat*, amely tartalmazza az összes értelmezett konstans, eljárás és függvény nevét és a szükséges paraméterek típusát.

Mivel, véleményünk szerint, egy áttekinthető és egyszerű program sokkal többet nyújt mindenféle magyarázatnál, az új ismereteket programok segítségével mutatjuk be, ahogyan ezt eddig is tettük. Ennek a módszernek további előnye, hogy a közölt programok átalakításával, bütykölésével egyszerű lehetőség nyílik a kipróbálásra. Lássuk tehát a programokat!

Első példánk igen egyszerű; a GRAPH.P hívásának bemutatása mellett mindössze egyetlen grafikus eljárást — a körrajzolást a **circle** eljárás segítségével — alkalmaz. A program listája a *46. ábrán* látható, maga a program a cikk fejlécében lévő képet állítja elő.

A második program rajzok készítését segíti. A képernyőn megszerkesztett rajzok egyébként nyomtatóra is kivihetők a DOS GRAPHICS.COM fájljának segítségével.

Eddigi ismereteinkhez képest a programban két újdonság szerepel. Az egyik a **case..of** szerkezet, amely többirányú programelágazást valósít meg; említését az eddigiekben elkerültük, mert problémáink megoldásához elegendő volt az **if..then,** illetve az **if..then..else.** Ugyancsak újdonság az a két grafikus eljárás — a **getpic** és a **putpic** —, amely a grafikus képernyő egészének, illetve egy részének elmentését vagy visszaállítását valósítja meg. A teljes képernyő tárigénye 16000+6 bájt. Ez a két eljárás nem okvetlenül szükséges ugyan pont ebben a programban, de alkalmazásuk bemutatására a program szinte felkínálta önmagát.

A **case..of** utasítás pontos szintaxisa a következő:

```
case kifejezés of
  konstans1,konstans2,...:utasítás1;
  konstans3,konstans4,...:utasítás2;
  ...
end;
```

Azaz ha a kifejezés kiértékelésekor az eredmény az egyes konstansokkal egyező értékű, az adott sorban lévő utasítás ke-

```
const
  North = 0;
  East  = 90;
  South = 180;
  West  = 270;

procedure Graphics;                                          external 'GRAPH.BIN';
procedure GraphMode;                                         external Graphics[0];
procedure GraphColorMode;                                    external Graphics[3];
procedure HiRes;                                             external Graphics[6];
procedure HiResColor(Color: Integer);                        external Graphics[9];
procedure Palette(N: Integer);                               external Graphics[12];
procedure GraphBackground(Color: Integer);                   external Graphics[15];
procedure GraphWindow(X1,Y1,X2,Y2: Integer);                 external Graphics[18];
procedure Plot(X,Y,Color: Integer);                          external Graphics[21];
procedure Draw(X1,Y1,X2,Y2,Color: Integer);                  external Graphics[24];
procedure ColorTable(C1,C2,C3,C4: Integer);                  external Graphics[27];
procedure Arc(X,Y,Angle,Radius,Color: Integer);              external Graphics[30];
procedure Circle(X,Y,Radius,Color: Integer);                 external Graphics[33];
procedure GetPic(var Buffer; X1,Y1,X2,Y2: Integer);          external Graphics[36];
procedure PutPic(var Buffer; X,Y: Integer);                  external Graphics[39];
function  GetDotColor(X,Y: Integer): Integer;                external Graphics[42];
procedure FillScreen(Color: Integer);                        external Graphics[45];
procedure FillShape(X,Y,FillCol,BorderCol: Integer);         external Graphics[48];
procedure FillPattern(X1,Y1,X2,Y2,Color: Integer);           external Graphics[51];
procedure Pattern(var P);                                    external Graphics[54];
procedure Back(Dist: Integer);                               external Graphics[57];
procedure ClearScreen;                                       external Graphics[60];
procedure Forwd(Dist: Integer);                              external Graphics[63];
function  Heading: Integer;                                  external Graphics[66];
procedure HideTurtle;                                        external Graphics[69];
procedure Home;                                              external Graphics[72];
procedure NoWrap;                                            external Graphics[75];
procedure PenDown;                                           external Graphics[78];
procedure PenUp;                                             external Graphics[81];
procedure SetHeading(Angle: Integer);                        external Graphics[84];
procedure SetPenColor(Color: Integer);                       external Graphics[87];
procedure SetPosition(X,Y: Integer);                         external Graphics[90];
procedure ShowTurtle;                                        external Graphics[93];
procedure TurnLeft(Angle: Integer);                          external Graphics[96];
procedure TurnRight(Angle: Integer);                         external Graphics[99];
procedure TurtleDelay(Delay: integer);                       external Graphics[102];
procedure TurtleWindow(X,Y,W,H: Integer);                    external Graphics[105];
function  TurtleThere: Boolean;                              external Graphics[108];
procedure Wrap;                                              external Graphics[111];
function  Xcor: Integer;                                     external Graphics[114];
function  Ycor: Integer;                                     external Graphics[117];
```

rül végrehajtásra. Az **else** ágat azért tettük zárójelbe, mert alkalmazása nem szükségszerű (a megoldandó feladat típusa határozza meg, hogy használjuk-e vagy sem).

A **case..of** szerkezet a programban több helyen is előfordul, alkalmazása kézenfekvő volt; a billentyűzet figyelésére lényegesen egyszerűbb megoldást tett lehetővé, mint a sorozatosan egymásba épített **if..then.. else** utasítások.

A **getpic** és a **putpic** eljárások paramétereiről csak annyit, hogy a kép mentésekor a befoglaló téglalap két sarkának koordinátáit kell megadni, visszatöltéskor viszont az egyik sarok x és a másik sarok y koordinátáját. A program listája a *47. ábrán* látható. Egy, a programmal szerkesztett rajzot a *48. ábra* mutat be.

Következő programunk, amelynek listája a *49. ábrán* látható, a **fill..** eljárásokat mutatja be. Sem a program, sem az egyes eljárások nem igényelnek különösebb magyarázatot, mindössze annyit tartunk szükségesnek megjegyezni, hogy a **fillshape** csak akkor festi ki az alakzatot, ha a **bordercolor** szín azonos a megrajzolt zárt idom (példánkban a kör) körvonalának színével.

Végezetül utolsó programunk (lásd az *50. ábrán* látható listát) az ún. turtle (teknősbéka) grafikai eljárások és függvények alkalmazását szemlélteti. A programban szerepel a **$U+** direktíva is; ezzel az volt a célunk, hogy a meglehetősen lassú — vagy lelassított — program futását ne kelljen végigvárni, hanem az bármikor leállítható legyen. A táblázat és a komment sorok segítségével a program megérthető, ezért különösebb magyarázatot nem is fűzünk hozzá.

Nagy Imre

47. ábra

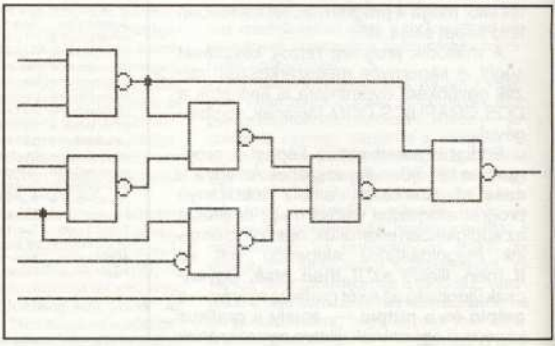
```
program grafika_segito;

{$I graph.p  }

label ujra;
var ch:char;
    i,x,y,x1,y1,radius:integer;
    kep:array[1..16006] of byte;

procedure kurzorpozicio;
begin
     gotoxy(37,1);
     writeln(x:3);
     gotoxy(37,2);
     writeln(y:3)
end;

procedure kurzor;
begin
     case ch of
          'b':x:=x-1;
          'j':x:=x+1;
          'f':y:=y-1;
          'l':y:=y+1
     end;
     kurzorpozicio;
end;

procedure ujvonal;
begin
repeat
      plot(x,y,0);
      kurzor;
      kurzorpozicio;
      plot(x,y,1);
      read(kbd,ch);
until ch='+'
end;

procedure kor;
begin
     x1:=x;y1:=y;
     gotoxy(1,1);
     write('radius=');
     readln(radius);
     write('b,j,f,l?');
     read(kbd,ch);
     case ch of
          'b':x1:=x1-radius;
          'j':x1:=x1+radius;
          'f':y1:=y1-radius;
          'l':y1:=y1+radius
     end;
     circle(x1,y1,radius,1);
     gotoxy(1,1);
     writeln('              ');
     writeln('              ');
     ch:=' '
end;

begin {főprogram}
```

```
     clrscr;
     textcolor(2);
     writeln('  Rajzoló program');
     writeln;
     textcolor(7);
     writeln('Kezdőpont: x=160,y=100');
     writeln('Kurzormozgatás:');
     write('  b=balra, j=jobbra, ');
     writeln('f=fel, l=le');
     writeln('Rajzolás kikapcsolása: -');
     writeln('Rajzolás bekapcsolása: +');
     writeln('Kör rajzolása: c');
     writeln('A rajzolt kör helyzete:');
     write('  b,j,f,l ');
     writeln('a kurzorhoz képest');
     write('  minden más billentyű: ');
     writeln('a középpont a kurzor');
     write('Kép kimentése és törlés  ');
     writeln('bekapcsolt rajzolásnál: k');
     write('Kilépés bekapcsolt ');
     writeln('rajzolásnál: ESC');
     textcolor(2);
     writeln;write('Programinditás: ');
     writeln('bármely billentyű');
     read(kbd,ch);
     x:=160;y:=100;
     graphcolormode;
     gotoxy(35,1);writeln('x=');
     gotoxy(35,2);writeln('y=');
     palette(3);
     plot(x,y,1);
     kurzorpozicio;
     ujra:read(kbd,ch);
     if ch=#27 then halt;
     if ch='k' then
     begin
          getpic(kep,0,0,319,199);
          clearscreen;
          gotoxy(1,1);
          writeln('Rajz vissza: k');
          repeat
               read(kbd,ch);
          until ch='k';
          putpic(kep,0,199);
     end;
     if ch='-' then ujvonal;
     if ch='c' then kor;
     kurzor;
     plot(x,y,1);
     goto ujra;
end.
```

46. ábra

```
program demo;
{$I graph.p  }
var x,y,x1,y1,radius:integer;
var ch:char;
begin
     repeat
     graphcolormode;
     for x:=1 to 12 do
     for y:=1 to 8 do
     circle(24*x,20+5*y,22,1);
     for x:=0 to 1 do
     for y:=0 to 2 do
     begin
          gotoxy(5+17*x,4+2*y);write('PASCAL GRAFIKA');
     end;
     gotoxy(1,25);
     write('Vege:ESC,Folytatas:barmely mas billentyu');
     read(kbd,ch);
     until ch=#27;
     textmode;
end.
```

48. ábra



```
program graphp2;

{Teknosbéka-grafika program}

{$I graph.p  }
{$U+}

var ch:char;
    i,j:integer;

procedure koordinatak;
{A teknöc koordinátáinak kiírása}
begin
     gotoxy(36,1);
     writeln(xcor:3);
     gotoxy(36,2);
     writeln(ycor:3)
end;

begin
     graphcolormode;
     palette(2);
     turtledelay(30);
{A teknöc mozgásának késleltetése}
     showturtle;
{A teknöc kijelzése}
     setheading(0);
{A rajzolás szögének beállítása}
     for j:=1 to 60 do
     begin
          koordinatak;
          turnright(90);
{Jobbra fordulás a megadott szöggel}
          forwd(2*j);
{Előre a megadott távolságra}
          setpencolor((j mod 3)+1);
{A rajzolás színének beállítása}
          if (j mod 2)=0 then penup
          else pendown
{A toll felemelése vagy lerakása}
     end;
     gotoxy(1,1);
     writeln('Torles: barmely billentyu');
     read(kbd,ch);
     gotoxy(1,1);
     writeln('                          ');
     pendown;
     turtledelay(0);
     setpencolor(0);
     for j:=60 downto 1 do
          begin
          koordinatak;
          back(2*j);
{Vissza a megadott távolságra}
          turnleft(90)
{Balra fordulás a megadott szöggel}
     end
end.
```

50. ábra

49. ábra

```
program graphp1;
{$I graph.p  }
var ch:char;
begin
     graphcolormode;
     palette(3);
     fillscreen(3);
{Az ernyó átszinezése}
     read(kbd,ch);
     fillpattern(100,10,220,80,1);
{A megadott téglalap kifestése}
     read(kbd,ch);
     circle(160,100,40,2);
     read(kbd,ch);
     fillshape(150,100,2,1);
{A kör kifestése}
end.
```

CAPS

Feladatmegoldás számítógéppel

Nem hiszem, hogy akadna olyan kisdiák, aki még ne álmodott volna a házi feladatot helyette megíró, mindentudó robotról. A számítógép lehetőségeit ismerők tudják, hogy egy ilyen elképzelés nem tartozik a mesék világába. Minden a programon múlik.

A leckeírás tökéletes programjához olyan hatalmas tudásbázisra van szükség, amely a jelenlegi házi számítógépek adattárolóiban nem férne el. Ezért csak egy szűkebb feladat programjának elkészítésére vállalkozhatunk, de látni fogjuk, hogy így is nagy terhet veszünk le a megoldó válláról. A redukált célt fejezi ki a program és a benne alkalmazott módszer neve is: CAPS, amely a Computer Aided Program Solving (számítógéppel támogatott problémamegoldás) rövidítése. Ebből is kitűnik, hogy a tervezett program nem önállóan, automatikusan készíti el a matematikai, fizikai vagy más számítási feladat megoldását, hanem segíti a feladatot megoldani szándékozót a megoldás keresésében.

Alapelvek

A feladatmegoldás számítógépes támogatásának alapgondolatához a következő megfigyelés vezetett. A feladat megoldását kereső agymunkája két, lényegesen különböző tevékenység váltakozó végrehajtásából áll:

1. Mozgósítja tudásbázisát, azaz ismereteinek tárházában, emlékezetében keres a feladathoz hasonlót, olyat, amelynek megoldását ismeri. Ha ilyen analóg feladatot nem talál, akkor olyan tételt keres, amelyet a konkrét feladat megoldásában vagy legalább a feladat egy részletének megoldásában felhasználhat. A siker két dologtól függ. Egyrészt a megoldó ismereteinek mennyiségétől, másrészt a megoldásban szerzett gyakorlatától. Az elsőhöz segítségül lehet hívni közbülső tudásbázist: lexikonokat, tankönyveket és nem utolsósorban kézikönyveket, képletgyűjteményeket. Mint a bevezetőben említettük, ezt az információtömeget el lehetne helyezni egy számítógépes adatbázisban, de erre a házi mikrogépek tárolókapacitása kicsinek bizonyulna. A feladatmegoldási gyakorlatot természetesen semmivel nem lehet pótolni.

2. A megoldó másik típusú tevékenysége valójában nem tekinthető szellemi munkának, annak ellenére, hogy ezt is az agyával végzi. Ez lényegében adminisztráció. A feladat megoldása közben fel- vagy meg kell jegyeznie azokat a képleteket, összefüggéseket, amelyeket a feladat egyes részeinek megoldásánál alkalmazni akar. Másrészt a megoldás keresése közben arra is figyelnie kell, hogy az összegyűjtött részmegoldások rendszere teljes és ellentmondásmentes legyen. Nyomon kell követnie a megoldásmenet állását: van-e még kidolgozatlan részlet, messze van-e a végcél, az új részfeladat vajon a kész megoldás irányába viszi-e a munkát, és így tovább. Végül a részkérdések mindegyikének megoldása után a megoldónak fel kell építenie a részletekből a megoldás tervét, a számítás algoritmusát. Mindezek a tevékenységek a megoldás keresése közben elvonják a megoldó szellemi energiáinak egy részét az előző, valóban szellemi munkától.

A CAPS program ez utóbbi tevékenységet vállalja magára.

A feladat elemzése

A program ismertetése előtt nézzük meg, hogyan dolgozik a feladat megoldója. Egy nagyon egyszerű példát választunk illusztrációként, hangsúlyozva, hogy ennél lényegesen bonyolultabb feladatok megoldására is alkalmas a program. Az *1. ábrán* egy falra szerelt forgódaru vázlata látható. (Geometriai feladatok gyűjteménye, 1567. feladat.) Ki kell számítani a BC rúd hosszát. Ismerjük a következő szakaszokat: MC=5, AB=4,2 és AC=13.

Első dolgunk az ismeretlen és az adatok jelölése, nevesítése, melyet célszerű az ábrán is bejelölni (lásd az ábra jobb oldalán). A megoldás közben szükségünk lehet további segédismeretlenekre, melyekről a feladat szövege nem szól. Ezeket menet közben nevesítjük és jelöljük az ábrában.

Az előkészületek után kezdődhet az igazi megoldás. Elsőként a következő kérdésre keresünk választ: hogyan számítható ki az ismeretlen. Ez a kérdés nem azonos a feladatban megfogalmazottal, hiszen ott az a feltétel is szerepel, hogy az ismeretlent a megadott adatokból kell kiszámítani. A feladat megoldója tehát szabad kezet kap arra, hogy az ismeretlen kiszámításához milyen mennyiségeket, adatokat és/vagy segédismeretleneket kíván (tud) felhasználni. Példánkban az MBC derékszögű háromszögből Pitagorász tételét (tudásbázis!)

felhasználva kapjuk a megfelelő képletet:

$x=\sqrt{u^2+a^2}$

ahol u egy új segédismeretlen. Ne tévesszen meg a mintapélda egyszerűsége. Gondoljunk olyan összetett feladatra, amelynek megoldása közben 10—15 segédismeretlent is fel kell használni. Ezért a megoldó jól teszi, ha a felírt képletet azonnal végignézi, és felírja a jobb oldalon szereplő változókat:

```
u=...
a=...
```

Ezzel az egyszerű fogással nagy lépést tettünk a megoldáshoz vezető úton. Így ugyanis automatikusan megfogalmaztuk a következő megoldandó részfeladatokat. Példánkban ezek: hogyan számítható ki u, illetve a? Az előbbire egy lehetséges válasz az ábra jelölései szerint:

```
u=h—b
h=...
b=...
```

A második kérdésre egyszerű a felelet:

```
a=5,
```

hiszen ez a változó a feladat egyik adatát jelöli: a=MC. A válaszban ilyenkor nem keletkezik újabb nyitott kérdés. Az előző válaszban viszont még van adósságunk. Ismét ötletre van szükség, amely most is a jó öreg Pitagorászhoz vezet el:

$h=\sqrt{c^2-a^2}$

```
c=...
```

Végül a két nyitott kérdésre a válaszok:

```
b=4,2
c=13
```

Ezzel a feladatmegoldás első szakaszának, az analízisnek a végére értünk: minden ismeretlenhez, segédismeretlenhez megadtunk egy olyan képletet, amelynek segítségével azokat a többi változóból — legyenek adatok vagy segédismeretlenek — ki tudjuk számítani. A megoldás következő szakaszának, a szintézisnek a példánk szerinti elemzése előtt nézzük meg a CAPS programnak azt a részét, amely az analízist végzi.

Az analízis algoritmusa

A program az adattömbök dimenziójának megadása után első lépésben a feladat ismeretlenjeit kéri a felhasználótól (140–160-as sorok). A programot olyan feladatok megoldására is alkalmassá tettük, amelyekben egynél több a kiszámítandó ismeretlenek száma. Ezeket a kezelőnek egybeírva, elválasztójelek nélkül kell megadnia. A program a v$ változóban gyűjti a feladat változóit.

Az adminisztrációt a program 200—390-es sorai végzik. Először kiválasztjuk a v$ láncból a feladat i-edik változóját, és ezt a vi$-be töltjük. A képernyőre a vi$-t és egy = jelet kiírva bekérjük ennek az ismeretlennek a kiszámítására alkalmas képletet. A programot felkészítettük arra az esetre is, amikor a felhasználó nem tudja a részfeladatot megoldani. Ekkor a képlet helyett egy ? jelet kell beírnia, mire a program megáll. Ne felejtsük el, gondolkodnia a megoldónak kell, a program „csak" adminisztrál.

Ha a beírt képlet, amely a kepl$(i)-be került, valódi válasz, akkor végigmegyünk a karakterein (280—380), és megkeressük a benne szereplő változókat. A 290—340-es sorokban kódolt vizsgálatokból látható, hogy „változónak" olyan karaktert tekintünk, amely egyedül álló betű. Ez azt jelenti, hogy a CAPS program használója nem alkalmazhat a programozási nyelvekben megszokott több betűből — vagy betűkből és számjegyekből — álló neveket. Ugyanakkor használhatók a több betűből álló függvénynevek. Sőt, mivel a program ebben a változatában a függvények nevét nem ellenőrzi, használhatók a BASIC-ben ismeretlen matematikai függvények vagy a BASIC-től eltérő jelölések. Például: arcsin, tg, ctg, lg, log2, ln, gy (az sqr helyett), kgy (köbgyök) és így tovább. A beírás formáját csupán az köti meg, hogy a képletet billentyűzeten írjuk, tehát nem írhatunk kitevőt, törtvonalat, gyökjelet. Ezek helyett a programnyelvekben alkalmazott a↑2 vagy (a+2)/(b—3) jelölésekkel tehetjük egyértelművé formuláinkat. Egyre azonban ügyelni kell: nem használhatók egyetlen betűből álló vagy egy betűt és más jeleket tartalmazó függvénynevek, mert ezeket a program a feladat egy változójának tekinti és megzavarodik.

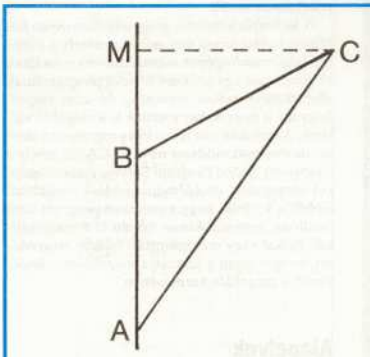
A program 350-es sorára akkor kerül a vezérlés, ha a képlet p-edik karaktere (y$) betű, és előtte (x$), illetve utána (z$) nem betű áll. Ebben a sorban ellenőrizzük, hogy ez a betű szerepel-e a változókat gyűjtő v$ láncban. Ha nem (360-as sor), akkor csatoljuk v$S végére, és n-et, a változók számlálóját növeljük. A 370-es sorban álló műveletről később szólunk.

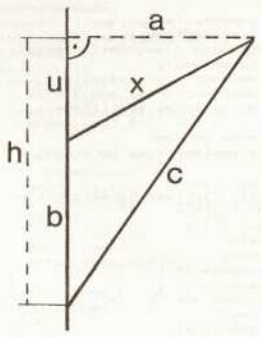
A képleteket bekérdező ciklust (220—390) addig ismételjük, amíg minden változóra rá nem kérdeztünk. A példánk megoldása közben képzett v$ láncot és a beírt kepl$(i) képletsort az 1. ábra bal alsó részében láthatjuk.

Közvetett hivatkozások

Térjünk most vissza példánkhoz, és vizsgáljuk meg a megoldás második szakaszát, a szintézist.

1. ábra





Közvetlen hivatkozások

	x	u	a	h	b	c
x		*	*			
u				*	*	
a						
h			*			*
b						
c						

Közvetett hivatkozások

	x	u	a	h	b	c
x		*	*	*	*	*
u			*	*	*	*
a						
h			*			*
b						
c						

i	v$	kepl$(i)
1	x	x=sqr(u↑2+a↑2)
2	u	u=h-b
3	a	a=5
4	h	h=sqr(c↑2-a↑2)
5	b	b=4.2
6	c	c=13

A megoldás

```
 1 a=5
 2 b=4.2
 3 c=13
 4 h=sqr(c↑2-a↑2)
 5 u=h-b
 6 x=sqr(u↑2+a↑2)
```

Ebben a szakaszban az összegyűjtött képletek, részmegoldások alapján kell a megoldási tervet megszerkeszteni. Azokban a feladatokban, amelyeknek a megoldásához a CAPS programot fel akarjuk használni, a számítási algoritmus szerkezete egyszerű. Nem tartalmaz sem ciklusokat, sem elágazásokat. Ezért a megoldási terv elkészítéséhez nem kell mást tenni, mint a képleteket megfelelő sorrendbe rendezni.

Példánkban ezt a sorrendet nagyon egyszerű megállapítani, de a sok változót igénylő feladatban már komoly gondot okozhat. A kiutat egy táblázat bevezetése adja, melyet a programban t%(n,n)-nel jelöltünk. Ennek a táblázatnak minden sora és oszlopa a feladat egy-egy változójának felel meg. A logikai értékekkel feltöltött táblázat t%(i,j) eleme akkor és csak akkor igaz=—1, ha az i-edik változó kiszámítására megadott képletben szerepel a j-edik változó. Ezt a táblázatot a közvetlen hivatkozások táblázatának nevezzük.

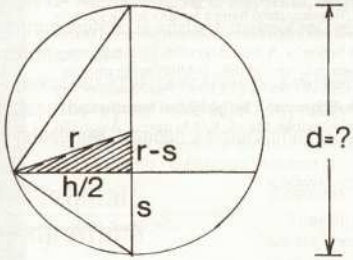
A program a képletek bekérdezése közben a 370-es sorban tölti fel a táblázat megfelelő elemeit. A példánkban készült táblázat az 1. ábrán bal oldalt látható. A közvetlen hivatkozások táblázata azonban nem elegendő a megoldási terv megszerkesztéséhez. Előfordulhat ugyanis, hogy a feladat megoldója olyan képletet (egyenletet) ad meg, amelynek jobb oldalán is szerepel a bal oldalon álló változó.

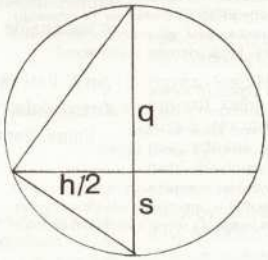
Az is megeshet, hogy több képletben kölcsönös hivatkozások szerepelnek. Ezért a programban a t%(n,n) táblázatot átalakítjuk úgy, hogy a t%(i,j) logikai értékek a két változó közötti közvetett hivatkozásokat adják meg. A közvetett hivatkozások táblázatát a 400—510-es sorokban, k=1,2,...,n lépéses rekurzióval állítjuk elő. A rekurzió k-adik lépésében a t%(i,j) elemet akkor állítjuk igaz=—1 értékre (460-as sor), ha már a (k—1)-edik lépésben igaz volt, vagy t%(i,k) és t%(k,j) együtt igaz volt. A rekurzió végén tehát t%(i,j) akkor lesz igaz, ha az i-edik változó kiszámításához a j-edik változót közvetlenül vagy akármelyik másik közvetítésével fel kell használni.

A közvetett hivatkozások táblázatából már meg lehet állapítani, hogy vannak-e kölcsönös hivatkozások a képletekben. A kölcsönös kapcsolatban álló változóknak megfelelő sorokban a főátlóban t%(i,j)=igaz elemek állnak. Ezt vizsgáljuk a program 530—570-es soraiban. Ha találunk ilyen elemet, akkor az 1000-es sorban kezdődő részre ugrunk, ahol kiírjuk az összes képletet, megjelölve közöttük a kritikusakat. Ekkor a felhasználó dönthet, hogy újra kezdi-e a megoldást vagy feladja.

Ilyen sikertelen, majd megismételt megoldási kísérletet mutatunk be a *2. ábrán*. A feladat egy kör átmérőjének (d) kiszámítása, ha adott egy körszeletének húrja (h=200) és a körszelet magassága (s=25). Az első megoldásban a besatírozott háromszögre alkalmaztuk Pitagorász tételét. A második megoldásban a Thálesz-tételt és a derékszögű háromszög magasságára vonatkozó mértaniközép-tételt használtuk fel.

2. ábra





Közvetlen hivatkozások

	d	r	h	s
d		*		
r		*	*	*
h				
s				

Közvetett hivatkozások

	d	r	h	s
d		*	*	*
r		*	*	*
h				
s				

```
* = rossz

  d=2*r
* r=sqr((h/2)↑2+(r-s)↑2)
  h=200
  s=25

kezdjük újra? (i/n)
```

Közvetlen hivatkozások

	d	q	s	h
d		*	*	
q			*	*
s				
h				

Közvetett hivatkozások

	d	q	s	h
d		*	*	*
q			*	*
s				
h				

i	v$	kepl$(i)
1	d	d=q+s
2	q	q=(h/2)↑2/s
3	s	s=25
4	h	h=200

```
A megoldás
  1 s=25
  2 h=200
  3 q=(h/2)↑2/s
  4 d=q+s
```

A megoldás terve

Nézzük most azt az esetet, amikor a képletek rendszere nem tartalmaz hurkot. A program 600—750-es soraiban állítjuk össze a feladat algoritmusát a következőképpen. A közvetett hivatkozások táblázatában az adatoknak — és csak ezeknek — megfelelő sorokban csupa hamis=0 érték szerepel. Nyilván a számításnak azzal kell kezdődnie, hogy az adatokat jelölő változókba behelyettesítjük azoknak a feladatban megadott értékét. A program 630—690-es soraiban megkeressük a t%(n,n) táblázat egy ilyen i-edik sorát. A 700—710-es sorokban kiírjuk a számítási lépés sorszámát (k) és a megtalált i-edik változóhoz tartozó kepl$(i) formulát. Ez a formula az adatoknál egy konstansérték-adás, az ismeretleneknél egy valódi képlet. A program 720—740-es soraiban a feldolgozott változónak megfelelő oszlopot és sort töröljük a táblázatból. Az oszlopot a hamis=0 értékekkel töltjük fel, jelezve, hogy a hivatkozás (ha egyáltalán volt) erre a feldolgozott változóra a továbbiakban közömbös. A táblázat i-edik sorát viszont igaz=—1 értékre állítjuk, hogy a következő üres sorok keresésénél ne választhassuk ki újra.

Az összes adat kiválasztása után azoknak az ismeretleneknek a sora lesz üres (hamis), amelyeknek a kiszámításához csak az adatokra van közvetlenül szükség. Ezeket ugyanúgy dolgozzuk fel, mint előbb az adatokat. Mivel a t%(n,n) táblázatban úgynevezett lineáris keresést alkalmazunk, előfordulhat, hogy egy-két adat feldolgozása után egy ismeretlent választ ki a program. Nyilván akkor, ha ez a táblázatban megelőzi a

többi adatot, és kiszámításához csak az előzőleg kiválasztott adatok szükségesek. Ez azonban nem érinti az algoritmus helyességét.

Mint jeleztem, a bemutatott példák nagyon egyszerű feladatok, így azt a téves elképzelést sugallják, hogy a megoldástervben a képletek a beírással ellentétes sorrendben jelennek meg. Ez azonban ritka eset. Aki a CAPS programot elkészíti és rendszeresen használja, nagyon hamar találkozni fog ellenpéldákkal.

A figyelmes olvasó talán már észrevette a rendszernek egy olyan szolgáltatását, amelyet a program céljának megfogalmazásakor elhallgattunk. A sikeres megoldás végén a program a feladat megoldásának algoritmusát a képernyőre írja az egyes lépések sorszámával együtt. Ha a felhasználó a képletek megadásakor szigorúan csak BASIC függvényeket és műveleti jeleket használt, akkor ez a megoldásterv formailag azonos egy BASIC programmal, pontosabban a feladatot megoldó BASIC programmal. Nincs más dolgunk, mint ugyanezeket a képernyőn látható sorokat a BASIC szerkesztővel egyenként közölni. A teljes képernyős szerkesztővel dolgozó gépek használói még egyszerűbb helyzetben vannak. Nem kell mást tenniük, mint egy NEW parancs kiadása után a képernyő tetejére vinni a kurzort és n-szer lenyomni a RETURN (ENTER) billentyűt. Az így szerkesztett nyers programból „igazi" úgy készülhet, hogy az adatokat egy INPUT listába helyezzük, és a program végére egy, az ismeretleneket kiíró PRINT-et csatolunk.

Fejlesztés

A CAPS program itt közölt változata szándékos egyszerűsítéssel készült, annak érdekében, hogy az érdeklődők az alkalmazott eljárást jobban megismerjék, valamint azért, hogy ne csak egyetlen géptípus tulajdonosai foglalkozhassanak a megvalósítás gondolatával. Az egyszerű lemásolás mellett célszerű, sőt helyenként szükséges némi kiegészítés a program biztos működéséhez. A közölt változat a C16 és Plus/4 BASIC-jében (CBM V 3.5) készült. A C64-en futó CBM V2 verzióhoz képest a DO—LOOP szervezésű ciklust, az INSTR(.) függvényt és a USING utasítást használjuk. Ezek a Simon's BASIC-ben megtalálhatók, de némi programozási gyakorlattal kiválthatók. Más géptípusra átíráskor az említett utasítások hiányában ugyanezt a módszert kell alkalmazni. A program listájában néhány PRINT utasításban a Commodore-tulajdonosok által jól ismert vezérlőkódok szerepelnek. Ezek jelentése a következő:

[clr]=a képernyő törlése
[hom]=a kurzort a bal felső sarokba visszük
[rvs]=inverz kiírás kezdete
[cud]=kurzor egy sorral lejjebb

A programban szerepel négy olyan sor — 410, 430, 470 és 490 —, amelyeket a belőtt programból el lehet hagyni (a sorok a közvetett hivatkozások előállítása közben kiírják a táblázat változó elemeit). Ez a rész elég időigényes, hiszen már 10 változó esetén is 1000-szer kell a 460—470 ciklusmagot végrehajtani. A program 110-es sorában adjuk meg a használható változók számát. Ennyi változó esetén még valószínűleg elfér a képernyőn a megoldási terv, hacsak nincsenek nagyon hosszú képletek. Az m maximálisan 26-ra változtatható, hiszen ennyi betű van az angol ábécében. Ha ez a szám kevésnek bizonyulna, akkor a program 290—340-es soraiban vizsgált feltételeket kell úgy átdolgozni, hogy kisbetűk és nagybetűk egyaránt használhatók legyenek változók jelölésére.

A kiegészítésnek véleményem szerint az adatbevitelek ellenőrzésére, az n > m kivédésére és a felhasználót tájékoztató dialógus megszervezésére mindenképpen ki kell terjednie. Ezek megvalósítása után kerülhet sor egyéni ötletek kivitelezésére: grafika a feladat ábrájának előállítására, figyelmeztető hang a kezelői hibák jelzésére, felhívás a beírt üzenet megerősítésére (Biztos vagy benne?). A legérdekesebb fejlesztési lehetőséget már említettük: BASIC program szerkesztése. A C16-os és a Plus/4-es használói egy ilyen változathoz a kereskedelmi forgalomban is hozzájuthatnak Matek-S.O.S. néven. Az önálló megoldásra vállalkozók számára még ajánlható a beírt képletek szintaktikus ellenőrzésének megszervezése is.

Dr. Hack Frigyes

```
100 rem ***** c.a.p.s. ******************
110 m=20              : rem"Változók max.
120 dim t%(m,m)       : rem"Táblázat
130 dim kepl$(m)      : rem"Képletek
140 rem"Az ismertelenek megadása
150 input "[clr]ismeretlenek :";v$
160 n=len(v$)  : if n=0 then 150
200 rem"*** A képletek beírása *********
210 i=0
220 do while i<n
230   i=i+1
240   vi$=mid$(v$,i,1)
250   print vi$"=";  : input kepl$(i)
260   if kepl$(i)="?" then end
270   kepl$(i)=vi$+"="+kepl$(i)
280   for p=3 to len(kepl$(i))
290     y$=mid$(kepl$(i),p,1)
300     if y$<"a" or y$>"z" then 380
310     x$=mid$(kepl$(i),p-1,1)
320     if x$>="a" and x$<="z" then 380
330     z$=mid$(kepl$(i),p+1,1)
340     if z$>="a" and z$<="z" then 380
350     j=instr(v$,y$)
360     if j=0 then n=n+1 : v$=v$+y$
370     t%(i,instr(v$,y$))=-1
380   next p
390 loop
400 rem"*** A kozvetett hivatkozások ***
410 print"[clr][rvs]"v$
420 for k=1 to n
430   print"[hom]"
440   for i=1 to n
450     for j=1 to n
460       t%(i,j)=t%(i,j) or t%(i,k) and t%(k,j)
470       print using "#";abs(t%(i,j));
480     next j
490     print
500   next i
510 next k
530 rem"*** Ellentmondások keresése ****
540 i=0
550 do until i=n or t%(i,i)
560   i=i+1
570 loop
580 if t%(i,i) then 1000 :rem"Ellentmondó
590 :
600 rem"*** A megoldás levezetése ******
610 print "[clr]";
620 for k=1 to n
630   i=0
640   do
650     i=i+1 : j=0
660     do until j=n or t%(i,j)
670       j=j+1
680     loop
690   loop while  t%(i,j)
700   print using " ## ";k;
710   print kepl$(i)
720   for j=1 to n
730     t%(j,i)=0 : t%(i,j)=-1
740   next j
750 next k
760 end
1000 rem"*** A hibák kiírása ***********
1010 print"[clr][rvs]* = rossz[cud]"
1020 for k=1 to n
1030   if t%(k,k) then h$="* ":else h$="   "
1040   print h$; kepl$(k)
1050 next k
1060 print"[cud] kezdjuk ujra? (i/n)"
1070 getkey r$
1080 if r$= "i" then run
1090 end
```

FELADATOK — MEGOLDÁSOK

Sorozatunkat elsősorban középiskolásoknak szánjuk, de reméljük, hogy minden olvasónknak tanulási lehetőséget és szórakozást nyújt.

A feladatok a Nemes Tihamér országos számítástechnikai verseny színvonalának felelnek meg. Minden esetben olyat választunk, amely röviden, gyorsan megoldható, de a megoldáshoz ötletre van szükség. A megoldást mindig a következő számban közöljük.

Mivel változatosságra törekszünk, különböző programozási nyelveket használunk. Az is előfordul, hogy egy feladatra több programnyelven is közlünk megoldást, ezzel is elősegítve az ismeretszerzést.

A szerkesztőség várja az olvasók, a versenyzők leveleit. A legötletesebb program beküldőjét könyvutalvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

17. feladat: Anagramma

Írjon olyan programot, amely megadott betűkből a betűk felcserélésével előállítható összes szót kilistázza, és a listában mindegyik szó csak egyszer fordul elő! A program a betűket csak egy példányban tárolhatja.

Megoldás

Mielőtt a feladat megoldásához hozzáfognának, számoljuk ki, hogy az N betűből hányféle szó rakható össze! Tételezzük fel először, hogy az összes betű különböző.

Ha csak egy betűnk van (például A), akkor abból csak egyféle szó képezhető (A). Ha van eggyel több betűnk (B), akkor azt az előző elé vagy mögé tehetjük, s így két szót képezhetünk (AB és BA). A harmadik betűt (C-t) már három helyre tehetjük, és így 2×3 megoldást kaphatunk (CAB, ACB, ABC, CBA, BCA, BAC). Az N-edik betűt már N helyre tehetjük, és így 1×2×3...×N, matematikai jelöléssel N! (faktoriális) -féleképpen rakhatjuk sorba.

Ha a betűk nem különbözők, akkor az ily módon kapott megoldások között több megegyező lesz. Ha K darab megegyező betű van, akkor ez K!-ad részére csökkenti a megoldások számát, mivel K betű K!-féleképpen rakható sorba. Ha több egymással megegyező karakter van (például AABCCC), akkor a megoldások száma:

$$\frac{6!}{2! \cdot 3!} = 60$$

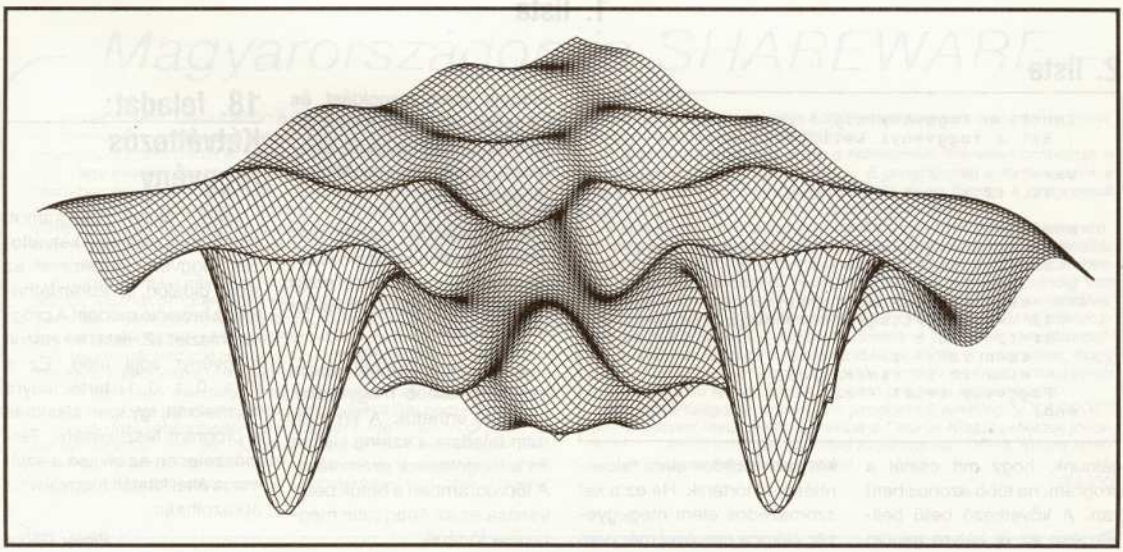
Ennek a számításnak, amit most elvégeztünk, két haszna is van: egyrészt ötletet meríthetünk a gondolatmenetből a program megírásához, másrészt a kapott megoldásokat könnyen ellenőrizhetjük: a program előállította az összes anagrammát, ha mindegyik szó különböző, és számuk megegyezik a számítottal.

Ezek után nézzük a Turbo Pascal 5.0 nyelven IBM PC-re írt programot *(1. lista)*!

Az anagrammák generálását az Anag rekurzív szubrutin végzi. Paraméterként megkapja, hogy az anagstr sztring hányadik betűjénél tartunk. Ezután a következő (st-edik) betűt rendre beilleszti a sztring végére, majd az azt megelőző helyekre. Minden esetben, ha ez nem a legutolsó, rekurzívan hívja önmagát a következő betű elhelyezéséhez. Ha ez a legutolsó, akkor rekurzív hívás helyett a megoldás kiírása következik.

Miután az összes lehetséges helyre elhelyeztük a következő betűt, az eredeti sztringet helyre kell állítani, majd következhet az ezt megelőző betűk következő permutációjának előállítása.

Már csak azt kell megvizs-



```
program anagramma;

uses
  Crt;

var
{ Az anagramma string:}
  anagstr : string;
{ Segedvaltozo: }
  c : char;
{ Szamoljuk a permutaciokat: }
  ancount : integer;


procedure WriteAn(var st : string);
{ Kiirja a stringet. }

begin
  Write(ancount:5,':',st,
        '':(10-Length(st)));
  ancount := ancount+1;
end;


procedure Anag(st : integer);
{ Rekurziv generalo rutin. }
{ A string st-dik betujet visszafele
  haladva beilleszti a elozoek koze
  az osszes lehetseges helyre.
  Ha st nem a string utolso betuje
  a kovetkezo betut rekurziv hivassal
  illeszti be.
  Ha st az utolso betu, akkor a kapott
  megoldasokat kiirja. }

var
  i : integer;
  c : char;
  elso : integer;

begin
  i:=st;
  elso:=1;
  repeat
    if st=Length(anagstr)
    then { Ez egy megoldas:}
      WriteAn(anagstr)
    else { Meg nincs vege:
           rekurziv hivas:}
      Anag(st+1);
    if i>1
    then begin

      { Ha ket azonos betut talal
        akkor kilep }
      if (anagstr[i] = anagstr[i-1])
      then begin
        elso:=i;
        i:=0
      end

      { Egyebkent felcsereli
        a ket betut. }
      else begin
        c := anagstr[i];
        anagstr[i] := anagstr[i-1];
        anagstr[i-1] := c;
      end;
    end;

    { Elozo helyre lep: }
    i:=i-1;
  until i < 1;

{ Mielott kiszallna visszarendezi a
  stringet. }
  c := anagstr[elso];
  for i:=elso+1 to st
  do anagstr[i-1] := anagstr[i];
  anagstr[st] := c;
end;


begin
  repeat
    ClrScr;
    Writeln('Anagramma  (Pinyo V1.0)');
    repeat
      Writeln('Add meg a stringet: ');
      Readln(anagstr);
    until Length(anagstr)>0;
    ancount:=1;
    Anag(1);
    c := ReadKey;
  until false;
end.
```

1. lista

2. lista

```
function Fuggveny(x,y : real):real;
{ Ezt a fuggvenyt kell kirajzolni. }

var
  f1, f2 : real;

begin
  f1 := 30*sqrt(sqr(x-0.4)+sqr(y-0.9));
  if f1 < 1E-5
    then f1 := 1
    else f1 := sin(f1)/f1;
  f2 := 30*sqrt(sqr(x-0.8)+sqr(y-0.4));
  if f2 < 1E-5
    then f2 := 1
    else f2 := sin(f2)/f2;
  Fuggveny := f1 + f2;
end;
```

gálnunk, hogy mit csinál a program, ha több azonos betű van. A következő betű beillesztése az új helyre mindig két szomszédos elem felcserélésével történik. Ha ez a két szomszédos elem megegyezik, akkor a cserével már nem kapunk újabb megoldást, és az ezt követő helyekre sem érdemes már ezt a betűt elhelyezni, mert azok enélkül is előállnak majd.

Ilyen esetben tehát már csak az eredeti sztring visszarendezése van hátra, majd következik a visszatérés az előző szintre.

A program további részei már különösebb magyarázat nélkül is érthetők. A WriteAn rutin feladata a sztring kiírása és a megoldások számolása. A főprogramban a betűk beolvasása és az Anag rutin meghívása történik.

18. feladat: Kétváltozós függvény

Írjon olyan programot, amely kirajzolja egy kétváltozós függvény grafikonját az előző oldalon, az *ábrán* láthatóhoz hasonló módon! A programrészlet *(2. lista)* a kirajzolt függvényt adja meg. Ez a (0..1, 0..1, 0..1) tartományra normalizált, így igen alkalmas a program tesztelésére. Természetesen az olvasó a saját maga által kitalált függvényt is ábrázolhatja.

Pintér Gábor

ZAK McKRACKEN and the alien mindbenders III.

Bolyongjunk tovább, amíg egy olyan terembe érünk, amelyben egy nagy térkép látható a falon, és tőle jobbra, a szfinx rajza alatt néhány újabb jel. Kapcsoljunk Zakra, és menjünk a szfinx bal lábához. Ide rajzoljuk le a Marson az előbb látott rajzot, és lépjünk be a keletkezett ajtón. Ezentúl a termeken keresztül mindig olyan ajtón haladjunk tovább, amelyen egy ember és a napkorong látható. Menjünk egészen addig, míg egy olyan terembe érünk, ahol az egyik ajtó fölött két szem látható. Menjünk be, kapcsoljunk át Annie-ra, és utazzunk vele Egyiptomba ugyanebbe a szobába. Olvastassuk el Annie-val a gombok felett a hieroglifákat, és a megfelelő sorrendben nyomjuk meg a gombokat. Rajzoljuk le Zakkal az előtűnő térképet a már meglevőre, mert csak így tudunk a Marsra is teleportálni. A gombok felett látható egy újabb jel, jegyezzük meg ezt is.

Teleportáljunk Zakkal a Marsra, és keressük meg vele a középső ajtó mögötti labirintusban az ankhot. Ballagjunk át a jobb szélső ajtó mögötti terembe, és helyezzük az ankhot az ott lévő panelba. Most kikapcsol az erőtér, és hozzáférhetünk az aranykulcshoz. Vigyük mindhárom szereplőt a nagy terembe, ahol Zak tegye fel az oxigénmaszkot és az akváriumot, és szigetelje le az akváriumot a szigetelőszalaggal (a szerszámosláda tárgyai között található). Melissát és Leslie-t vigyük az űrrepülőhöz, és töltsük fel a szkafanderüket levegővel. Ezután mindhármukat vigyük a monolithoz, és mindegyikükkel vegyünk két jegyet. Szálljanak be mindhárman a villamosba. A piramisnál Leslie-vel seperjük el a port a bejárat elől, Zak pedig nyissa ki a csipesszel az ajtót. Vonuljunk be, és míg Leslie-vel nyomjuk a szarkofág lábát, Zakot és Melissát vigyük fel a fenti terembe. Rakjuk be a kulcslyukba az aranykulcsot, és álljunk közel a kristályhoz. Melissával nyomjuk meg a gombot, és Zakra kapcsolva vegyük fel gyorsan a fehér kristályt. (Lehet többször is próbálkozni.) Ezután teleportáljuk Zakot az egyiptomi piramisba, és állítsuk úgy a falon lévő kart, hogy Annie is be tudjon jönni. Miután Annie megérkezett, adjuk oda neki mind a három kristályt, és helyezzük a talpazatra a világító tárgyat, arra a gyertyatartót és a gyertyatartó száraira a három kristályt. Ezután kapcsoljuk fel a két oldalt lévő kapcsolókat.

Ezzel Zak teljesítette küldetését. Az idegenek továbbállnak, és az emberiség megmenekül. Zak tetteiért Nobel-díjat kap, és mivel mindezt könyvben is megírja, elnyeri a Pulitzer-díjat is. Egyébként házasságot kötnek Annie-val.

Remélem, mindenki élvezni fogja ezt az igen bonyolult, de szórakoztató játékot.

Tass Csaba

Magyarországon is SHAREWARE

Kedves olvasó! Talán már ön is került olyan helyzetbe, amikor egy adott feladat megoldására keresett felhasználói programot, de a reklámanyagból nem derült ki, hogy az adott szoftver megfelel-e a célnak. Vagy esetleg nem tudott dönteni két program között, mert a leírásokban mindkettőt a legjobb, legszebb, leghatékonyabb stb. jelzőkkel illették, s hátrányaikat bölcsen elhallgatták a gyártók. Valószínűleg ön is gyorsan meg fog barátkozni egy új számítógépes fogalommal, a shareware-rel, s két társával, a public domainnel és a freeware-rel.

A public domain név olyan programfajtákat jelez, amelyek a fejlesztők hasznos programozási melléktermékeinek tekinthetők. Általában kis terjedelmű részprogramok, rutinok ezek, amelyek szabadon felhasználhatók, saját programokba beépíthetők.

A nálunk már eddig is használt, ún. szabad programok tartoznak a freeware kategóriába, amelyek szabadon másolhatók, terjeszthetők, de gyártójuk fenntartja magának valamennyi szerzői jogot (a programot megváltoztatni, dokumentációját átírni nem engedi).

A legérdekesebb kategória a shareware. Ide olyan programok tartoznak, amelyek már nem sokban különböznek a kereskedelmi forgalomban kapható szoftverektől, terjesztésükre speciális szoftverkönyvtárak szakosodtak. Ezek a szoftverkönyvtárak csak a terjesztéssel összefüggő minimális költségeket (a lemez ára, csomagolási, terjesztési költségek) számítják fel.

A shareware program révén a felhasználó fillérekért próbálhat ki nagyteljesítményű programokat is. E programmal a forgalmazók a legjobb marketing-lehetőséget találták meg, hiszen a programok legjobban kipróbálásuk során ismerhetők meg.

A kereskedelmi forgalomban kapható programoktól a shareware-ek annyiban különböznek, hogy a gyártók ezt nem azok helyettesítésére, hanem azok reklámozására hozták forgalomba. Hozzátehetjük: minimális költséggel. A shareware programoknak mindig van valamilyen hiányosságuk. Nincsenek megfelelően dokumentálva, találhatók bennük „fékek", amelyek „lebutítják" a terméket, hiányoznak belőlük bizonyos — többnyire fontos — részletek, de előfordulhat például az is, hogy a nyomtatott lapra kiírják a programok, hogy „próba" vagy „demo". Ezzel a gyártók mintegy rákényszerítik a vevőt a kereskedelmi forgalomban kapható program megvételére.

Magyarországon a shareware programok jelenleg SOLARSOFT fantázianéven kerülnek forgalomba a Cédrus Kisszövetkezet jóvoltából. A Cédrus szoftverkönyvtára folyamatosan bővül, főként amerikai és nyugatnémet termékekkel.

(SANY)

PILLANATFELVÉTEL A PIACRÓL

Sokszor készült már olyan áttekintés, amely megkísérelte összegyűjteni, mi, mikor, mennyibe kerül a számítástechnikai termékeket árusító boltokban. Mi is elindultunk egy nyári napon barátommal, hogy ha nem is a teljes kínálatot, de legalább annak egy részét feltérképezzük. Öt üzletet kerestünk fel.

Elsőként a Metró-Comp szaküzletbe látogattunk el. Itt azonban kellemetlen meglepetés ért minket: kijelentették, hogy nem hajlandók nyilatkozni, mégpedig azért, mert ehhez engedélyt kell kérniük, amit jelen pillanatban nem áll szándékukban beszerezni. Ezek után kitessékeltek az üzletből. Én azt hiszem, hogy ez nem a legjobb reklám a cégnek, és ahol az érdeklődőkkel így bánnak, ott a vevőkkel sem foglalkozhatnak különbül.

Ezután a Metró-Print üzletbe mentünk. Itt érdekes módon készségesen adtak felvilágosítást, pedig ugyanúgy a Skálához tartozik, mint a Metró-Comp!! Az itt látottakat az *1. táblázatban* foglaltuk össze. A Metró-Print egyébként nemcsak számítástechnikával foglalkozik, hanem műszaki cikkeket, hanglemezt, kazettát is árul. Az érdeklődőt felkészült szakemberek várják, és készségesen adnak felvilágosítást, akár az árukészletről, akár bármilyen szakmai kérdésről van szó. A táblázat alapján elmondhatjuk, hogy ez a bolt inkább a hardverre szakosodott.

METRÓ-PRINT

A termék neve	Árusításának ideje	Ára
Márkátlan 5 1/4" floppylemez	jelenleg kapható	80 Ft/db
PARROT 5 1/4"	jelenleg kapható	130—149 Ft
3M 5 1/4" high density	jelenleg kapható	250—390 Ft
IBM PC alapkiépítés	jelenleg kapható	150 000 Ft
COMMODORE 64	időszakonként	16 000 Ft
1541 floppy	időszakonként	18 500 Ft
COMMODORE Plus/4	nincs adat	10 000 Ft
Datasette Plus/4-hez	nincs adat	3000 Ft/db
botkormány mikrokapcsoló	jelenleg kapható	1500 Ft/db

1. táblázat

Harmadiknak a Novotrade PC szalonjába látogattunk el. Ez az üzlet a PC-k és tartozékaik árusítására szakosodott. A kínálatot a *2. táblázatban* láthatjuk.

A szakemberek szerint (külföldiek!) ez a szalon egész Európában egyedülálló. Valóban, mi is csak a legjobbakat mondhatjuk el róla. Bő a hardver- és szoftverkínálat, és nem utolsó dolog, hogy az érdeklődő kellemes körülmények között, kényelmesen kipróbálhatja a kiválasztott terméket. Sok, PC-ről szóló szakkönyvet is láttunk.

Sétánk következő, negyedik állomása a Novotrade 2C üzlete volt *(3. táblázat)*. Itt is jó tapasztalatokat szereztünk. A kiválasztott programot a vásárló nyugodt körülmények között betöltheti az ott található számítógépek egyikébe és kipróbálhatja. A kisgépekre talán itt a legnagyobb a programkínálat. Sok érdekes könyvet is láttunk, igaz, elég borsos áron.

Az ötödik és egyben utolsó helyszínünk a Budafoki úton található, számítástechnikai termékek árusítására szakosodott Ápisz szaküzlet volt. Az itteni árakat a *4. táblázatban* láthatjuk. Itt a hardverkínálat elég szegényes, viszont sok szoftvert és rengeteg féle festékszalagot láttunk. Itt kaphatók egyébként számítástechnikával foglalkozó újságok is, többek között a Mikroszámítógép Magazin is.

Összefoglalva elmondhatjuk, hogy elég jó a számítógépkínálat, igaz, különböző dolgokért különböző üzletekbe kell menni: például programokért a Novotrade 2C-be és az Ápiszba, hardverért a Metró-Printbe és a Novotrade 2C-be, számítástechnikai papíráruért az Ápisz boltba, PC-kkel kapcsolatos vásárlás esetén pedig a PC szalont érdemes felkeresni.

Persze sokan foglalkoznak számítógép forgalmazásával a meglátogatott üzleteken kívül is. Ezért legközelebb néhány újabb helyszín kínálatáról számolunk be.

Bácsi Péter

NOVOTRADE PC SZALON

A termék neve	Ára
XT-kompatibilis konfiguráció, a Dataplantól (640 kbájt RAM, 360 kbájtos lemezegység, 20 Mbájtos merevlemezegység, monokróm monitor)	145 600 Ft
XT-kompatibilis konfiguráció, a Microsystemtől (640 kbájt RAM, 360 kbájtos lemezegység, 20 Mbájtos merevlemezegység, monokróm monitor)	116 000 Ft
AT-kompatibilis konfiguráció, a Dataplantól (640 kbájt RAM, 1,2 Mbájt lemezegység, 20 Mbájt merevlemezegység, monokróm monitor)	187 000 Ft
AT-kompatibilis konfiguráció az Ázsiótól (10 MHz, 640 kbájt RAM, 1,2 Mbájt lemezegység, monokróm monitor, soros/párhuzamos csatoló	99 000 Ft
IBM PC 386-AT-kompatibilis konfiguráció az ACP-től (20 MHz, 2 Mbájt RAM, 1,2 Mbájt lemezegység, 80 Mbájt merevlemezegység, monokróm monitor, párhuzamos csatolókártya	368 000 Ft
MICROTEK MS-300C átfutásos scanner a Novotrade-től	243 000 Ft
Genius GM-6 egér a Digit -től	9 000 Ft
Manager egér, infravezérelt	24 500 Ft
Houston EDMP-61 plotter (A4, 1 toll, 0,025 felbontás) a Microsystemtől	799 000 Ft
Monokróm monitor, 12"-os, zöld	12 000 Ft
Színes monitor, 14"-os	36 800 Ft
1,2 Mbájt hajlékonylemez-egység az Econorgtól	55 000 Ft
360 kbájt lemezegység	40 000 Ft
Fujitsu DL 5600 mátrixnyomtató	285 000 Ft
Star Laser-8 lézernyomtató a Cobrától (8 lap/perc, 1 Mbájt RAM, 300 000 lap élettartam)	399 000 Ft
40 Mbájtos, beépíthető streamer a Műszertechnikától	120 000 Ft
60 Mbájtos, beépíthető streamer a Műszertechnikától	150 000 Ft
20 Mbájtos, beépíthető merevlemez a Mikroszerviztől	85 000 Ft
60 Mbájtos, beépíthető merevlemez a Mikroszerviztől	130 000 Ft

2. táblázat

NOVOTRADE 2C

A termék neve	Árusításának ideje	Ára
COMMODORE 64	jelenleg kapható	16 000 Ft
1541 drive	időszakonként	18 500 Ft
C=1230 printer	időszakonként	23 000 Ft
Festékkazetta MPS 1230-hoz	időszakonként	500 Ft
Deltex Plus Modul C Plus/4-hez		3 700 Ft
BASIC 7.0 cartridge Plus/4		1 940 Ft
Datasette C 64-hez		2 500 Ft
Játékprogramok kazettán	jelenleg kapható	340—580 Ft
Játékprogramok lemezen	jelenleg kapható	450—780 Ft
ENTERPRISE 128	jelenleg kapható	11 900 Ft
ENTERPRISE-hoz 3 1/2 drive	jelenleg kapható	18 500 Ft
SPECTRUM emulátor	jelenleg kapható	7 770 Ft
PARROT 3 1/2	jelenleg kapható	3000 Ft/cs
PARROT 5 1/4	jelenleg kapható	800 Ft/cs
50 db-os floppy lemeztartó	jelenleg kapható	1 200 Ft
PARROT 96 sávos	jelenleg kapható	1800 Ft/cs

3. táblázat

ÁPISZ

A termék neve	Árusításának ideje	Ára
WABASH 5 1/4	jelenleg kapható	1890 Ft/cs
PELIKAN 5 1/4	jelenleg kapható	2490 Ft/cs
3M 5 1/4	jelenleg kapható	2000 Ft/cs
HIGH DENSITY 5 1/4	jelenleg kapható	3500—5900 Ft
3 1/2	jelenleg kapható	590—750 Ft
PELIKAN floppy-tisztító	jelenleg kapható	1 730 Ft
COMMODORE 64	időszakonként	16 000 Ft
1541/II	időszakonként	18 500 Ft
ROBOTRON-C64 interfész	jelenleg kapható	15 400 Ft

4. táblázat

Biztos betöltés

ENTERPRISE

A Nick koprocesszor olyan grafikus lehetőségekkel árasztja el a felhasználót, amelyeket igazán csak képszerkesztővel lehet kiaknázni. A legismertebb ilyen a PaintBox mellett a Lorigraph, amelyhez sajnos nem mellékeltek betöltő szoftvert. Így rajzainkat nem használhatjuk fel saját munkáinkban. Ezen segít a *listán* látható program.

A Lorigraph a kimentett kép elé egy olyan 13 bájtos adatcsomagot rögzít, amely a lap összes tulajdonságát hordozza. Ennek felépítése a következő:

Bájtsorszám	Funkció
0.	Video Mode (1,5)
1.	Video Color (0—3)
2.	Video X (1—42)
3.	Video Y (1—99)
4—11.	A paletta alsó színei
12.	Bias, a felső színek

Munkám első fele ezekből az adatokból kreál egy grafikus lapot, ez a rész önmagáért beszél. Némi magyarázatra a 43-as sortól szorul a program. Az „S" gépi kódú rutin meghatározza, tárolja a lap méretét, és visszaadja a BASIC-nek a Nick-féle lapkezdőcímet mint 16 bites előjeles számot. Ebből megállapítható a videoszegmens száma, a 43—56-os sorokban ettől függően ágazik el a végrehajtás. A kellő paraméterekkel hívott „F" eljárás kiszámolja és raktározza a Z80-as címeket. Az illető szegmensszámmal paraméterezett „R" rutin a következőket végzi:

- letiltja a megszakításokat,
- belapoz három egymást követő videolapot (max. 48 kbájt) úgy, hogy az első mindig a CPU első lapján lesz,
- az operációs rendszer 6-os funkciójával elvégzi a töltést,
- visszaállítja a felhasználói lapkiosztást, lezárja a háttércsatornát,
- engedélyezi a megszakításokat.

Ez a rutin és szelektálás biztosítja, hogy a képadatok biztosan jó helyre kerüljenek.

Az eljárás hívása: CALL TUTTI („file-név", videocsatorna száma). A név tipikusan LORI3, de EXDOS esetén eltérő neveket kell használni. A VY-ban visszakapjuk a lap Y méretét, amely a displayhez kellhet.

Igazán szép, eredeti megoldás lenne, ha a betöltőt áthelyezhető rendszerbővítőként közölnénk, univerzálissá téve azt. Sajnos erre e cikkben terjedelmi okokból nincs lehetőség.

Az eljárást használva grafikát tölthetünk az EnterSPRITE-ok alá. Ehhez a Lorigraph-ban 16 színű, nagy felbontású, 40×20-as lapot és képet szerkesszünk. Ezt a 101-es videocsatornára kérjük.

Hajnal Csaba

```
10 PROGRAM "TUTTI.BAS"
11 ALLOCATE 80:LET VY=0
12 CODE T="????"
13 CODE R=HEX$("F3,1,B1,3,ED,78,F5,ED")
14 CODE =HEX$("69,C,2C,10,F7,3E,6A,ED")
15 CODE =HEX$("5B")&WORD$(T+2)
16 CODE =HEX$("ED,4B")&WORD$(T)
17 CODE =HEX$("F7,6,1,B3,3,F1,ED,79,D")
18 CODE =HEX$("10,FA,FB,3E,6A,F7,3,C9")
19 CODE S=HEX$("3E,65,6,3,F7,B")
20 CODE =HEX$("EB,B7,ED,42,22")
21 CODE =WORD$(T)&HEX$("60,69,C9")
22 DEF TUTTI(N$,CH)
23   NUMERIC H(0 TO 12)
24   OPEN #106:N$ ACCESS INPUT
25   FOR I=0 TO 12
26     GET #106:X$:LET H(I)=ORD(X$)
27   NEXT I
28   SET VIDEO MODE H(0)
29   SET VIDEO COLOR H(1)
30   SET VIDEO X H(2)
31   LET VY=H(3):SET VIDEO Y VY
32   POKE S+1,CH:SET BIAS H(12)
33   OPEN #CH:"VIDEO:"
34   FOR I=0 TO 7
35     SET #CH:COLOR I,H(I+4)
36   NEXT I
37   SELECT CASE VY
38   CASE 1 TO 27
39     DISPLAY #CH:FROM 1 TO VY
40   CASE ELSE
41     DISPLAY #CH:FROM 1 TO 27
42   END SELECT
43   SELECT CASE USR(S,0)
44   CASE 0 TO 16383 ! seg:252
45     CALL F(16384+USR(S,0))
46     CALL USR(R,252)
47   CASE 16384 TO 32767 ! seg:253
48     CALL F(USR(S,0))
49     CALL USR(R,253)
50   CASE -32768 TO-16385 ! seg:254
51     CALL F(49152+USR(S,0))
52     CALL USR(R,254)
53   CASE ELSE ! seg:255
54     CALL F(32768+USR(S,0))
55     CALL USR(R,255)
56   END SELECT
57   DEF F(X)
58     POKE T+2,REM(X,256)
59     POKE T+3,INT(X/256)
60   END DEF
61 END DEF
99 CALL TUTTI("LORI3",101)
```

TOP LISTA

	játék programok	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE
1.	Bard's Tale III.		●		●		●			●
2.	Out Run		●		●		●			●
3.	Sentinel	●								●
4.	TechnoCop		●		●		●			
5.	Jet Simulator	●								
6.	Riziko				●					
7.	Rocket Ranger		●	●	●					●
8.	Battle Chess	●	●							
9.	Zak McKracken		●		●					
10.	Stealth Fighter				●					●

	felhasználói programok	IBM	AMIGA	C-128	C-64	C+4(16)	SPECTR.	ENTERP.	TVC	APPLE
1.	Ventura Publish.	●								
2.	Foxbase 2	●								
3.	Geos 2.0				●	●				
4.	EasyCad 2	●								
5.	News Room	●			●					
6.	Art Studio 12				●		●			
7.	Amica Paint				●					
8.	PC-Write	●								
9.	Giga Cad +				●					
10.	Renegade				●					

Listánkat felhasználói, illetve játékprogramokból állítjuk össze. A legjobbakat, legérdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Enterprise-ra, TVC-re, Atarira és IBM-re készült programrangsorokat várunk havonta.

Címünk:
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
Diákszerkesztőség

BASIC-bővítések Commodore 16-ra

2. rész

Az előző részben már szó volt a tokenekről, tokenizálásról, detokenizálásról. Most a BASIC bővítésével foglalkozunk.

Nézzük meg a $0300—$0311-ig terjedő memóriarészt *(1. ábra)*. Ez az ún. BASIC-vektor tábla. Itt látunk három olyan vektort, amely a felhasználói tokenekhez, illetve parancsokhoz tartozik.

Lássuk, hogyan keresi ki a gép a BASIC-parancsokból azt, amelynek a tokenjét a programban tárolni kell! A $8A03 címen kezdődő szubrutin végzi el ezt a feladatot, és tokenizálja a BASIC-szavakat *(2. ábra)*. Láthatólag semmi más dolgunk nincs, mint egy olyan táblázatot készíteni a saját szavakból, ahogy azt az interpreter ismeri, majd megadva neki a saját táblázatunk kezdőcímét, kikeresi számunkra a tokent *(3. ábra)*.

Szinte ugyanez a tennivalónk akkor, ha detokenizálni akarunk. De ekkor a $8B40-nel kezdődő detokenizáló szubrutint kell meghívni *(4. ábra)*. Tudatjuk a szubrutinnal a saját BASIC-szavaink táblázatának kezdetét, majd a többit rábízhatjuk erre a szubrutinra *(5. ábra)*.

Most már csak a végrehajtás van hátra. Az interpreter rutinba beépítették azt a programrészt, ami a felhasználói tokennek a használatát könnyíti meg *(6. ábra)*.

Foglaljuk össze, mit kell tennünk ahhoz, hogy a felhasználói tokeneket használni tudjuk!

1. Táblázatba kell foglalni a saját BASIC-szavakat oly módon, hogy minden szó utolsó betűje shiftelve legyen beírva (ez fogja jelezni a szó végét): a táblázatot egy $00 bájttal kell befejezni (ez fogja jelezni a lista végét).

2. Táblázatot kell készíteni a saját parancsok kezdőcímeiről

1. ábra

```
$0300-$0301: $B686, BASIC meleginditás
$0302-$0303: $A7E7, egy sor beolvasása
$0304-$0305: $B93E, tokenizálás
$0306-$0307: $BB6E, listazás
$0308-$0309: $BBD6, utasitás értelmezés
$030A-$030B: $9417, kifejezést kiértékel

          felhasználói parancs
$030C-$030D: $B96A, tokenizálás
$030E-$030F: $BB88, listázás
$0310-$0311: $BC8B, végrehajtás
```

2. ábra

```
. 8a03  a9 81     lda #$81     BASIC
. 8a05  a0 8e     ldy #$8e     kulcsszavak
. 8a07  85 23     sta $23
. 8a09  84 22     sty $22
. 8a0b  a0 00     ldy #$00
. 8a0d  84 0b     sty $0b      token szám
. 8a0f  88        dey
. 8a10  c8        iny
. 8a11  20 a5 04  jsr $04a5    egy byte be
. 8a14  38        sec          hasonlitás
. 8a15  f1 22     sbc ($22),y  ha egyezik,
. 8a17  f0 f7     beq $8a10    következő
. 8a19  c9 80     cmp #$80     kulcsszó
. 8a1b  f0 1b     beq $8a38    egyezés
. 8a1d  b1 22     lda ($22),y
. 8a1f  30 03     bmi $8a24    szó vége
. 8a21  c8        iny
. 8a22  d0 f9     bne $8a1d
. 8a24  c8        iny          következő
. 8a25  e6 0b     inc $0b      token
. 8a27  18        clc          előző szó
. 8a28  98        tya          hosszát
. 8a29  65 22     adc $22      hozzáadja,
. 8a2b  85 22     sta $22      innen keres
. 8a2d  90 02     bcc $8a31    tovább
. 8a2f  e6 23     inc $23
. 8a31  18        clc
. 8a32  a0 00     ldy #$00
. 8a34  b1 22     lda ($22),y
. 8a36  d0 d9     bne $8a11    van még szó
. 8a38  05 0b     ora $0b
. 8a3a  85 0b     sta $0b
. 8a3c  60        rts
```

3. ábra

```
ass +4: pass 1 2
 2052                    .opt p
 2052 48        tokl     pha
 2053 a9 20              lda  #>nevtabla
 2055 a0 1e              ldy  #<nevtabla
 2057 20 07 8a           jsr  $8a07
 205a 68                 pla
 205b 90 d3              bcc  vege
 205d a5 0b              lda  bel
 205f 48                 pha
 2060 4c d6 89           jmp  $89d6
 2063 4c 6c 89 vege      jmp  $896c

2052-2066
```

4. ábra

```
. 8b66  20 d1 04  jsr $04d1   köv. byte
. 8b69  f0 50     beq $8bbb   ha 0, vége
. 8b6b  6c 06 03  jmp ($0306) DETOKEN
. 8b6e  10 df     bpl $8b4f   nem token
. 8b70  c9 ff     cmp #$ff    PI ?
. 8b72  f0 db     beq $8b4f   igen
. 8b74  24 0f     bit $0f     idézőjel ?
. 8b76  30 d7     bmi $8b1f   bekapcs.
. 8b78  c9 fe     cmp #$fe    USER token?
. 8b7a  d0 17     bne $8b93   nem
. 8b7c  c8        iny
. 8b7d  20 d1 04  jsr $04d1   köv. byte
. 8b80  f0 0c     beq $8b8e   ha 0, nincs
. 8b82  84 49     sty $49     tárolja
. 8b84  38        sec         kapcsoló=1
. 8b85  6c 0e 03  jmp ($030e) USER DETOK.
. 8b88  b0 c5     bcs $8b4f   ha kapcs=1
. 8b8a  a0 00     ldy #$00    nincs user
. 8b8c  f0 24     beq $8bb2   feltétlen
. 8b8e  88        dey               ugrás
. 8b8f  a9 fe     lda #$fe
. 8b91  d0 bc     bne $8b4f
. 8b93  aa        tax         token X-be
. 8b94  84 49     sty $49
. 8b96  a0 81     ldy #$81    táblázat
. 8b98  84 23     sty $23      elejének
. 8b9a  a0 8e     ldy #$8e    beolvasása
. 8b9c  84 22     sty $22     ($818e)
. 8b9e  a2 00     ldy #$00     tokenszám
. 8ba0  ca        dex          csökkentés
. 8ba1  10 0f     bpl $8bb2   ez az ?
. 8ba3  b1 22     lda ($22),y utasitás-
. 8ba5  48        pha          szöveg
. 8ba6  e6 22     inc $22     számláló
. 8ba8  d0 02     bne $8bac    növelése
. 8baa  e6 23     inc $23
. 8bac  68        pla
. 8bad  10 f4     bpl $8ba3   köv. byte
. 8baf  30 ef     bmi $8ba0   név vége

. 8bb1  c8        iny         utasitás-
. 8bb2  b1 22     lda ($22),y szöveg
. 8bb4  30 95     bmi $8b4b
. 8bb6  20 b2 90  jsr $90b2   kiirása
. 8bb9  d0 f6     bne $8bb1
. 8bbb  60        rts
```

(-1) a szokásos alsó-felső bájt sorrendben. Azért kell csökkenteni a címet, mert egy RTS gépi utasítással fogunk a BASIC-parancs kezdetéhez jutni, de az RTS után a mikroprocesszor a veremből elővett címhez egyet hozzáad *(7. ábra)*.

Természetesen, ha a táblázatokat összeállítottuk, a megfelelő szubrutinok kezdőcímeit a BASIC-vektorokban el kell helyezni.

Most lássuk azt a programot, amelyre minden BASIC-bővítésnél szükség van. Egy nagyon egyszerű utasítással, a COMPUTER-rel próbálom érzékeltetni, hogy a BASIC bővítése milyen könnyedén megvalósítható. Ennek az utasításnak a kiadásakor a gép kiírja a nevét *(8. ábra)*.

A programot begépelve, majd lefordítva, a következő parancscsal aktivizálhatjuk:

SYS 8192 <return

Írjuk be: COMPUTER <return
és a gép
válaszol: COMMODORE 16
READY.

Készítsünk egy egyszerű programot:

```
10 COMPUTER
20 END
```

Ha ezt a programot elindítjuk, a gép kiírja a saját nevét.

Nézzük meg, hogyan tárolta a számítógép ezt a programot *(9. ábra)*. Láthatjuk, hogy most apró változás van ott, ahol a tokennek kellene lennie. Egy token helyett kettő van, és az első SFE értékű, a második pedig S80. Ez azt jelenti, hogy a tárolt programban elhelyezkedő parancs az első felhasználói token (pontosabban a 0). Az általunk készített bővítés tehát teljesen beleilleszkedik az eredeti BASIC-be.

Kádár Sándor

5. ábra

```
2081 aa       detokl  tax
2082 84 49            sty  be2
2084 a0 20            ldy  #>nevtabla
2086 84 23            sty  be3
2088 a0 1e            ldy  #<nevtabla
208a 4c 9c 8b         jmp  $8b9c

2081-208d
```

6. ábra

```
. 8c3f  c9 fe     cmp #$fe    USER token?
. 8c41  f0 3f     beq $8c82   igen
. 8c43  c9 cb     cmp #$cb    tovább

. 8c82  20 73 04  jsr $0473   köv. byte
. 8c85  f0 06     beq $8c8d   ha 0, hiba
. 8c87  38        sec         kapcsoló
. 8c88  6c 10 03  jmp ($0310) USER vektor
. 8c8b  90 e4     bcc $8c71   user volt ?
. 8c8d  4c a1 94  jmp $94a1   ha nem,hiba
. 8c90  4c 7c 8e  jmp $8e7c   ugrás LET-
```

7. ábra

```
. 8c5e  0a        asl         cím kiszá-
. 8c5f  a8        tay               mítása
. 8c60  b9 84 83  lda $8384,y
. 8c63  48        pha         verembe
. 8c64  b9 83 83  lda $8383,y       tö|tés
. 8c67  48        pha
. 8c68  4c 73 04  jmp $0473   következő
                              byte
                              (és RTS a
                              címre)
```

8. ábra

```
pass +4: pass 1 2
2000                    .opt p
2000          tok       =    $030c

2000          detok     =    $030e
2000          vegre     =    $0310
2000          be1       =    $0b
2000          be2       =    $49
2000          be3       =    $23
2000          szovegki  =    $ff4f
                        ;
                        ;
2000 a9 1f              lda  #<tok1
2002 8d 0c 03           sta  tok
2005 a9 20              lda  #>tok1
2007 8d 0d 03           sta  tok+1
200a a9 33              lda  #<detok1
200c 8d 0e 03           sta  detok
200f a9 20              lda  #>detok1
2011 8d 0f 03           sta  detok+1
2014 a9 3f              lda  #<vegre1
2016 8d 10 03           sta  vegre
2019 a9 20              lda  #>vegre1
201b 8d 11 03           sta  vegre+1
                        ;
                        ;
201f 48         tok1    pha
2020 a9 20              lda  #>nevtab
2022 a0 4e              ldy  #<nevtab
2024 20 07 8a           jsr  $8a07
2027 68                 pla
2028 90 06              bcc  vege
202a a5 0b              lda  be1
202c 48                 pha
202d 4c d6 89           jmp  $89d6
2030 4c 6c 89 vege      jmp  $896c
                        ;
                        ;
2033 aa         detok1  tax
2034 84 49              sty  be2
2036 a0 20              ldy  #>nevtab
2038 84 23              sty  be3
203a a0 1e              ldy  #<nevtab
203c 4c 9c 8b           jmp  $8b9c
203f 38         vegre1  sec
2040 e9 80              sbc  #$80
2042 0a                 asl  a
2043 b9 58 20           lda  cimtab+1,y
2046 48                 pha
2047 b9 57 20           lda  cimtab,y
204a 48                 pha
204b 4c 73 04           jmp  $0473
                        ;
                        ;
204e 43 4f 4d nevtab    .asc "computeR"
2056 00                 .byt 0
                        ;
                        ;
2057 58 20      cimtabl .wor compu-1
                        ;
                        ;
2059 20 4f ff compu     jsr  szovegki
205c 43 4f 4d           .asc "commodo"
2063 52 45 2d           .asc "re-16"
2069 0d 00              .byt 13,0
206a 18                 clc
206b 60                 rts
2000-206c
```

9. ábra

```
>1000 00 08 10 0a 00 fe 80 00 :..... ..
>1008 0e 10 14 00 80 00 00 00 :........
>1010 00 00 00 00 00 00 00 00 :........
```

ABLAKOZÓ

Egy-egy saját felhasználásra vagy esetleg másoknak írt program sokkal barátságosabbá, emberközelibbé tehető, ha olyan felhasználói interfészt, menürendszert tudunk létrehozni, amely áttekinthetőbbé teszi a választási lehetőségek sorát, és megkönnyíti magát a választást. Ezeknek a feltételeknek az ablaktechnikát használó menürendszerek kielégítően eleget tesznek. A függetlenül felhasználható, a képernyő bármely pontjára helyezhető, tetszőleges méretű, színű és tartalmú ablakok segítségével és azok jó megválasztásával nagyon esztétikus képernyőképeket hozhatunk létre. Ennek a rendszernek az előnyei közé tartozik, hogy a választás módja egyértelmű, és a felhasználó hibás billentyűhasználatából eredő programhiba minimálisra csökkenthető. Ez főleg a BASIC-ben írt programoknál szokott probléma lenni.

Ez a program egyaránt használható BASIC-ből, assemblerből és bármilyen más nyelvből, amelyben lehetőség van arra, hogy a memóriába tetszőlegesen írhassunk és onnan olvashassunk. A kezdő felhasználók számára biztosan bonyolultnak és főleg nehézkesnek tűnik a sok POKE-olás, de a cél egy minden nyelven elérhető program létrehozása volt. Különben az új BASIC-utasítások létrehozásáról már sok cikk jelent meg, és azok alapján, esetleg némi segítség igénybevételével megírt rutinnal ez a hiányosság kiküszöbölhető. Az ablakok megtervezésének segítésére mellékeljük az *Ablakkreáló* nevű programot.

Felépítés

A program C000-tól kb. C400-ig terjedő területen helyezkedik el. Alapjában véve az egész 4 k-s (C000-D000) szeletet igénybe veszi, mivel az ablakok alatti képernyőfelületet és színinformációt saját maga után tárolja. Ez egyben megszabja a maximálisan egyszerre a képernyőn megjeleníthető ablakok területét. Mivel a rendelkezésre álló memória kb. 3 k, és egy karakter elmentése 2 bájtot igényel, ez maximum másfél képernyőnyi felületet jelent. Ez természetesen az ablakok számát nem maximálja.

Az ablakokat egy tetszőleges helyre kell előzetesen elhelyezni, és az első ablak memóriabeli kezdőcímét a C00D-C00E címek tartalmazzák. Ez az ablak fog az *Ablakozó* meghívásakor a képernyőre kerülni. Az ablakoknak alapvetően két fajtájuk lehet. Az elsőben a választható lehetőségek egymás mellett, a másodikban egymás alatt helyezkednek el (lásd az *1.* és a *2. ábrát*).

Az egyetlen kikötés, hogy — akár függőleges, akár vízszintes az ablak — a választható lehetőségek csak egy sorban, függőlegesen vagy vízszintesen helyezkedhetnek el. Ebből logikusan következik, hogy a vízszintes ablak csak 1 vagy 3 karakter magas lehet. (Ha 3 karakter magas, akkor az Ablakozó automatikusan úgy veszi, hogy az ablak első és harmadik sora a keret része, és csak a középső tartalmazza a lehetőségeket.)

„Deszka, üveg, gitt"

Minden ablak kötelező része a keret, ami viszont tetszőleges karakterekből állhat. Ez alól egyetlen kivétel az 1 karakter magas, vízszintesen elhelyezkedő ablak, ami általában a legfelső szintű ablak szokott lenni.

Az ablak definiálásához először is annak jellemzőit kell megadni, a következő sorrendben:

1. Az ablak bal felső sarkának x-koordinátája (0 <=x <=39)
2. Az ablak bal felső sarkának y-koordinátája (0 <=y <=24)
3. Az ablak szélessége
4. Az ablak magassága
5. Az ablakon lévő szöveg színe
6. A választható lehetőségek száma az ablakban.

A következő bájtok jelentős mértékben függenek a 6. pontban megadott számtól. Ugyanis minden egyes választható lehetőségnek meg kell adni az üzemmódját, választható további lehetőségeket vagy azt kínálva, hogy térjen vissza a hívó BASIC vagy más programba. Az újabb ablak esetén a kód „1", különben „0". Tehát a következő lépés a választható lehetőségek számával megegyező számú 1-es vagy 0-ás elhelyezése a tárban.

Már csak egy információtípus van hátra. Ez a választható lehetőségek számával megegyező számú címet jelenti, amely megadja, hogy a következő ablakot hol tároltuk (vagy visszatérés esetén az ugrási címet). Ezen a címen elhelyezhetjük saját gépi kódú programunkat, ami a megfelelő lehetőség választása esetén aktivizálódik. Ha nem akarunk ezzel a lehetőséggel élni, akkor egy olyan címet kell megadni, amin egy RTS utasítás áll.

Lehet, hogy ez így egy kicsit bonyolultnak tűnik, de a példát elolvasva könnyen érthetővé válik. Definiáljunk egy olyan ablakot, amelynek az 5,4 pozícióban van a bal felső sarka, 20 karakter széles, 5 karakter magas, az írásszín fehér, a választható lehetőségek száma 3, az első kettő újabb ablakokra mutat, amelyek a 4000 és 41AA címen vannak tárolva, a harmadik egy ugrás C00C-re, ahol egy RTS van! Ezt az ablakot a következő bájtsorozat definiálja:

ABLAKKREÁLÓ

```
1 REM ********************
2 REM *   ABLAKKREÁLÓ   *
3 REM *GRADONYI GERGELY*
4 REM ********************
10 DIM A#(20,20) :REM MAXIMUM 20 ABLAK
20 DIM A(20,20)  :REM MELYEK MAXIMUM 20
30 DIM F(20,20)  :REM SOROSAK ES MAS.
40 DIM M(20,5)   :REM 20 LEHETOSEGESEG
50 FOR F=0 TO 45
60 A#=A#+""
70 A#=A#+"_"
80 M$=M$+CHR$(32)
85 M$=M$+CHR$(157)
90 NEXT
100 PRINT "{DOWN}";DB;
105 PRINT ", ABLAK MERETEIT KEREM!"
110 INPUT "BAL-FELSO SAROK X-KORD";X
113 IF X<0 OR X>39 THEN 110
115 M(DB,0)=X
120 INPUT "BAL-FELSO SAROK Y-KORD";X
123 IF X<0 OR X>24 THEN 120
125 M(DB,1)=X
130 INPUT "ABLAK SZELESSEGE      ";X
133 IF X+M(DB,0)>40 THEN 130
135 M(DB,2)=X
140 INPUT "ABLAK MAGASSAGA       ";X
143 IF X+M(DB,1)>25 THEN 140
145 IF X<1 OR X>2 THEN 140
147 M(DB,3)=X
150 INPUT "BETUSZIN              ";X
153 M(DB,4)=X
155 IF M(DB,3)<>1 THEN 159
156 X#=LEFT$(A2#,M(DB,2)-2)
157 A#(DB,0)=CHR$(160)+X#+CHR$(160)
158 GOTO 190
159 X#=LEFT$(A1#,M(DB,2)-2)
160 A#(DB,0)=A#("+X#+"
163 X#=LEFT$(A4#,M(DB,2)-2)
165 A#(DB,2)=B1#+L#+X#+F
170 FOR F=1 TO M(DB,3)-2
173 X#=LEFT$(A2#,M(DB,2)-2)
175 A#(DB,F)=A#("+X#+"_F
180 NEXT F
190 PRINT "?"
200 FOR F=0 TO M(DB,3)-1
210 PRINT LEFT$(STR$(F+1)+".    ",4);
220 PRINT A#(DB,F)
225 PRINT LEFT$(A3#,LEN(A#(DB,F))+2);
227 INPUT A#(DB,F)
230 IF LEN(A#(DB,F))<>M(DB,2) THEN 210
240 NEXT F
250 INPUT "JO EZ AZ ABLAK ?(Y/N)";A#
260 IF A#="N" THEN 190
270 IF M(DB,3)=3 THEN 285
275 PRINT "E SORBAN A LEHETOSEGEK":
280 INPUT " SZAMA";M(DB,5) :GOTO 310
285 M(DB,5)=M(DB,3)-2
310 FOR F=1 TO M(DB,5)
320 PRINT "KOD MEG A";F;". LEHETOSEG";
325 INPUT " UZEMMODJAT";F(DB,F)
330 PRINT "KOD MEG E VALASZTAS UGRASI";
340 INPUT " CIMET";M(DB,F)
370 IF F(DB,F)>0 AND M(DB,F)=0 THEN A(DB,F)=45164
390 NEXT
500 INPUT "VAN MEG";A#
510 IF A#="I" THEN DB=DB+1:GOTO 100
520 PRINT "AKARSZ VALAMELYIK ABLAKON";
525 INPUT " VALTOZTATNI";A#
530 IF A#="N" THEN 600
540 PRINT "KOD MEG AZ ABLAK";
545 INPUT " SORSZAMAT";DB
550 GOTO 100
600 DB=0
700 PRINT "KOD MEG A TAROLAS";
705 INPUT " KEZDOCIMET";BA
710 IF DB=11 THEN END
720 IF M(DB,3)=0 THEN DB=DB+1:GOTO 710
725 PRINT "?"
730 FOR F=1 TO M(DB,3)
740 PRINT A#(DB,F-1)
750 NEXT
760 PRINT "EZT AZ ABLAKOT BERAKJAM";
765 INPUT " A MEMORIABA";A#
770 IF A#="N" THEN DB=DB+1:GOTO 710
780 FOR F=0 TO 5
790 POKE BA+F,M(DB,F)
800 NEXT
810 BA=BA+6
820 FOR F=1 TO M(DB,5)
830 POKE BA-1+F,F(DB,F)
840 NEXT
845 BA=BA+M(DB,5)
850 FOR F=1 TO M(DB,5)
855 X=256*INT(M(DB,F)/256)
860 POKE BA-2+F*2,M(DB,F)-X
870 POKE BA-1+F*2,INT(M(DB,F)/256)
880 NEXT
890 BA=BA+2*M(DB,5)
900 FOR F=1 TO M(DB,3)
910 FOR G=1 TO LEN(A#(DB,F-1))
920 PRINT "#";MID$(A#(DB,F-1),G,1)
930 POKE BA,PEEK(1024)
940 BA=BA+1
950 NEXT
960 NEXT
965 DB=DB+1
970 PRINT "DA MEMORIA KOVETKEZO URES";
975 PRINT " BAJTJA A      ";BA;"-ES"
980 INPUT "PARAMSZ UJ CIMET MEGADNI";A#
990 IF A#="N" THEN 710
1000 GOTO 700
```

5, 4, 20, 5, 1, 3, 1, 1, 0, 00, 40, AA, 41, 0C, C0

Ez a bájtsorozat írja le az ablak tulajdonságait, és ez után kell sorfolytonosan tárolni az ablak tartalmát. A karaktereket képernyőkódú alakban kell elhelyezni a memóriában. A vízszintes ablaknál minden egyes választható lehetőség között legalább egy SPACE-nek kell lennie, mivel az invertáló rutin az első SPACE-ig invertál. Ugyanebből az okból az első lehetőség előtt és az utolsó után szintén legalább egy SPACE-nek kell lennie. Ez a kikötés a keretes — azaz 3 karakter magas — ablakra is érvényes!

Ha kimegy az ajtón...

Van egy különleges üzemmód is az ablakok használatában. Ha egy akármilyen ablakban a lehetőségek számát 1-nek adjuk meg, az ablak képernyőre kerülése után a program elugrik a megadott címre. (Pontosan úgy, mint ha egy lehetőség kiválasztása történt volna.) Ez az üzemmód akkor hasznos, ha az ablakba beviteli lehetőséget akarunk bevinni, vagy ha egy szöveges üzenetet akarunk a képernyőre írni *(1. program)*. Ha már kiléptünk az Ablakozóból, akkor például BASIC-ból a SYS 49155 (C003) utasítással visszaléphetünk az utoljára megnyitott ablakba. Természetesen vigyázni kell, hogy közben ne rontsuk el az ablakok tartalmát, mivel azok nem fognak újból felíródni a képernyőre. Ha el akarjuk tüntetni az utolsó ablakot, a SYS 49158 (C006) utasítással megtehetjük. Általában érdemes eltüntetni az összes

```
A-VAR.   B-VAR.   C-VAR.   D-VAR.
```

1. ábra

```
A-VARIACIO
B-VARIACIO
C-VARIACIO
D-VARIACIO
E-VARIACIO
F-VARIACIO
G-VARIACIO
```

2. ábra

ablakot a választás után, mert ez jelentősen megkönnyíti a képernyő további használatát. A legjobb megoldás, ha a legfelső szintű menü a képernyő tetején egy sorban helyezkedik el, így csak arra kell vigyázni, hogy azt az egy sort ne írjuk át. Ha a képernyőn 80 karakter hosszú sorok maradnak (ez általában program írásakor fordul elő), akkor az ablakok egyes sorai eltűnhetnek. Ezért ajánlatos az Ablakozó meghívása előtt letörölni a képernyőt.

Egy menüprogram csak akkor lehet valóban hasznos, ha könnyen megtudhatjuk, hogy a felhasználó pontosan hol lépett ki a menürendszerből. Ebben a programban ennek egy elég egyszerű, minden nyelvből könnyen lekérdezhető megoldása szerepel. A 49170-es tárcímen található, hogy hány szintig jutott el a felhasználó. Az ezek után következő bájtok megadják, hogy az egyes szinteken melyik lehetőséget választotta. Vigyázni kell, mert a számunkra 1-es lehetőség a program számára a 0-ás, azaz egyet hozzá kell adnunk a beolvasott számhoz. Saját programunkban egy elvileg egyszerű — ámbár lehet, hogy terjedelmes — IF-es szerkezettel követhetjük a választások sorát *(2. program)*.

Az ablakokban a kurzormozgató billentyűkkel mozoghatunk, és a választást a RETURN leütésével jelezhetjük a gépnek. Ha egy ablakból vissza akarunk lépni az előzőbe, az F1 billentyűvel tehetjük meg. Ha a legelső ablakból visszalépünk, visszakerülünk programunkba, amely a hívó utasítás után fog folytatódni. Ugyanide kerülünk vissza a RETURN lenyomása után is. Logikusan ez után helyezkedik el a választást kiértékelő rutin.

Az Ablakkreáló

Ez a segédprogram hasznos lehet az ablakok létrehozásában, de az igazán kényelmes munkához egy monitorprogramra is szükségünk lehet. A program elég egyszerű a rövidség érdekében, de ha kiegészítjük lemezkezelő rutinokkal, ügyesebb szerkesztővel, esetleg automatikus ugrásicím-számlálóval, sokkal használhatóbbá tehetjük.

Elindítva az első ablak jellemzőit adhatjuk meg, majd magát az ablakot tervezhetjük meg.

ABLAKOZÓ

```
1 REM **********************
2 REM *     ABLAKOZO       *
3 REM *GARDONYI GERGELY*
4 REM **********************
10 BA=49152
20 FOR F=0 TO 7:FOR G=0 TO 7
30 READ A$:IF A$="VEGE" THEN END
40 A1=LEFT$(A$,1):A2$=RIGHT$(A$,1)
50 A1=ASC(A1$)-48:A2=ASC(A2$)-48
60 A1=A1+7*(A1>9):A2=A2+7*(A2>9)
70 POKE BA+G+F*8,A1*16+A2
80 C=C+A1*16+A2:NEXT G,F
90 READ A:IF A<>C THEN PRINT "HIBA" 100 C=0:BA=BA+64:PRINT"■":BA:GOTO 20
1000 DATA 4C,3B,C1,4C,AB,C1,4C,2F
1010 DATA C2,4C,6E,C0,68,00,48,00
1020 DATA 80,00,02,01,04,01,00,00
1030 DATA 00,00,00,00,00,00,00,00
1040 DATA 19,00,00,00,00,00,0A,04
1050 DATA 05,0A,8E,41,00,00,00,02
1060 DATA 04,0B,A9,FC,CE,FF,AD,2E
1070 DATA C0,91,FE,C8,AD,27,C0,91
1080 :
1090 DATA 4301
1100 :
1110 DATA FE,C8,AD,28,C0,91,FE,C8
1120 DATA AD,29,C0,91,FE,E6,FF,60
1130 DATA A0,FC,C6,FF,B1,FE,80,26
1140 DATA C8,C8,B1,FE,8D,27,C0,C9
1150 DATA B1,FE,8D,28,C0,C0,81,FE
1160 DATA 8D,29,C0,E6,FF,60,AD,2C
1170 DATA C0,D0,05,AD,2B,C0,D0,25
1180 DATA AD,12,C0,8A,48,B9,23,C4
1190 :
1200 DATA 10356
1210 :
1220 DATA 18,69,04,85,FE,B9,24,C4
1230 DATA 69,00,85,FF,AD,2C,C0,F0
1240 DATA 06,20,32,C0,4C,A7,C0,20
1250 DATA 50,C0,4C,A7,C0,AD,2A,C0
1260 DATA 85,FE,AD,2B,C0,85,FF,A9
1270 DATA 01,85,D7,20,B0,C0,C6,D7
1280 DATA AD,27,C0,85,D6,A0,26,C0
1290 DATA 85,D3,20,6C,E5,20,24,EA
1300 :
1310 DATA 8623
1320 :
1330 DATA A2,00,AC,26,C0,AD,2C,C0
1340 DATA F0,12,A5,D7,F0,07,B1,D1
1350 DATA 81,FE,4C,00,C1,B1,F3,81
1360 DATA FE,4C,00,C1,AD,2B,C0,D0
1370 DATA 12,A5,D7,F0,07,A1,FE,91
1380 DATA D1,4C,00,C1,A1,FE,91,F3
1390 DATA 4C,00,C1,A5,D7,F0,09,A1
1400 DATA FE,91,D1,AD,31,C0,91,F3
1410 :
1420 DATA 9529
1430 :
1440 DATA E6,FE,D0,02,E6,FF,C8,98
1450 DATA 38,ED,26,C0,C0,28,C0,90
1460 DATA B4,E6,D6,A5,D6,38,ED,27
1470 DATA C0,CD,29,C0,90,9C,A5,D7
1480 DATA F0,01,60,AD,28,C0,F0,01
1490 DATA 60,AC,12,C0,C8,98,0A,A8
1500 DATA A5,FE,99,23,C4,A5,FF,99
1510 DATA 24,C4,60,A0,00,8C,12,C0
1520 :
1530 DATA 9488
1540 :
1550 DATA C8,A9,39,8D,25,C4,A9,C4
1560 DATA 8D,26,C4,A0,0D,C0,85,A7
1570 DATA A0,0E,C0,B5,A0,8E,12,C0
1580 DATA A0,12,C0,A8,A5,A7,99
1590 DATA 07,C4,A5,A8,C8,99,07,C4
1600 DATA 20,5A,C2,A0,01,8C,2C,C0
1610 DATA B9,9C,20,C0,20,6E,C0,20
1620 DATA B5,C1,A0,00,8C,2C,C0,20
1630 :
1640 DATA 8108
1650 :
1660 DATA 6E,C0,4C,C3,C1,A0,80,01
1670 DATA A7,C9,02,D0,04,A9,01,D0
1680 DATA 02,A9,00,18,6D,2E,C0,8D
1690 DATA 2E,C0,65,A7,8D,2A,C0
1700 DATA A5,A8,69,00,8D,2B,C0,60
1710 DATA 4C,DE,C9,A0,12,C0,F0,F7
1720 DATA 0A,A0,B9,07,C4,85,A7,C8
1730 DATA B9,07,C4,85,A8,20,5A,C2
1740 :
1750 DATA 7949
1760 :
1770 DATA 4C,E5,C1,AC,2E,C0,88,0C
1780 DATA 30,C8,F0,2F,A9,00,8D,30
1790 DATA C0,AC,29,C0,C0,01,F0,0A
1800 DATA C8,03,F0,06,20,A4,C3,4C
1810 DATA E5,C1,20,54,C3,20,00,C2
1820 DATA A0,2F,C0,D0,08,20,2F,C2
1830 DATA AC,12,C0,F0,82,4C,E5,C1
1840 DATA AC,30,C0,B1,A7,F0,A9,A0
1850 :
1860 DATA 8658
1870 :
1880 DATA B0,F0,01,60,AC,12,C0,AD,
1890 DATA 30,C0,99,12,C0,0A,65,A7
1900 DATA 6D,2E,C0,8D,2A,C0,85,FE
1910 DATA A5,A8,69,00,8D,2B,C0,85
1920 DATA FF,A0,00,B1,FE,85,A7,C8
1930 DATA B1,FE,85,A8,4C,55,C1,AD
1940 DATA 12,C0,D0,01,60,A9,00,8D
1950 DATA 2C,C0,8D,2D,C0,20,6E,C0
1960 :
1970 DATA 9241
1980 :
1990 DATA CE,12,C0,AC,12,C0,B9,12
2000 DATA C0,8D,30,C0,98,0A,A8,B9
2010 DATA 07,C4,85,A7,C8,B9,07,C4
2020 DATA 85,A8,A0,00,B1,A7,8D,26
2030 DATA C0,C8,B1,A7,8D,27,C0,C9
2040 DATA B1,A7,8D,28,C0,C8,B1,A7
2050 DATA 8D,29,C0,C8,B1,A7,8D,31
2060 DATA C0,C8,B1,A7,8D,2E,C0,A5
2070 :
2080 DATA 9121
2090 :
2100 DATA A7,18,69,0C,90,02,E6,A8
2110 DATA 85,A7,60,AC,12,C0,8C,2F
2120 DATA C0,AC,29,C0,C0,01,F0,60
2130 DATA C0,D2,F0,5C,20,3E,F1,C9
2140 DATA 00,D0,01,60,C3,85,D0,8C
2150 DATA A9,00,8D,2F,C0,60,C9,11
2160 DATA F0,07,C9,91,D0,E6,4C,DA
2170 DATA C2,20,A4,C3,AC,2E,C0,88
2180 :
2190 DATA 8185
2200 :
2210 DATA 98,C0,30,C0,D0,8E,A9,80,
2220 DATA 8D,30,C0,20,A4,C3,4C,9C
2230 DATA C2,EE,30,C0,20,A4,C3,4C
2240 DATA 9C,C2,20,A4,C3,8D,30,C0
2250 DATA D0,0D,AC,2E,C0,88,8C,30
2260 DATA C0,20,A4,C3,4C,9C,C2,CE
2270 DATA 30,C0,20,A4,C3,4C,9C,C2
2280 DATA 20,3E,F1,C9,00,D0,01,60
2290 :
2300 DATA 8363
2310 :
2320 DATA C9,85,D0,06,A9,00,8D,2F
2330 DATA C0,60,C9,10,F0,25,C9,90
2340 DATA D0,E6,4C,15,C3,20,54,C3
2350 DATA AD,30,C0,D0,0D,AC,2E,C0
2360 DATA 88,8C,30,C0,20,54,C3,4C
2370 DATA F5,C2,CE,30,C0,20,54,C3
2380 DATA 4C,F5,C2,20,54,C3,AC,2E
2390 DATA C0,B9,9B,C0,3D,7D,D0,8D
2400 :
2410 DATA 8436
2420 :
2430 DATA A9,00,8D,30,C0,20,54,C3
2440 DATA 4C,F5,C2,EE,30,C0,20,54
2450 DATA C3,4C,F5,C2,A9,04,85,A8
2460 DATA A0,26,C0,18,69,01,AC,27
2470 DATA C0,D0,06,AE,29,C0,C6,F9
2480 DATA 11,AE,29,C0,C6,F0,01,C8
2490 DATA 18,69,28,90,02,E6,A8,85
2500 DATA D0,F0,85,A9,AE,30,C0,E8
2510 :
2520 DATA 8481
2530 :
2540 DATA B1,A9,C0,C9,20,F0,75,C8
2550 DATA C9,F0,D0,B1,A9,C0,C9,20
2560 DATA D0,F9,C0,00,F0,51,A5,C3
2570 DATA 20,F0,00,49,00,01,A9,C0
2580 DATA 4C,35,C3,A0,00,84,85,A9
2590 DATA 0D,20,C0,10,69,01,A0,20
2600 DATA 18,6D,27,C0,90,02,E6,AA
2610 DATA 18,69,01,90,02,E6,AB,18
2620 :
2630 DATA 8548
2640 :
2650 DATA 6D,30,C0,90,02,E6,AB,88
2660 DATA D0,E6,85,A9,AE,28,C0,CA
2670 DATA C6,A0,00,B1,A9,48,D0,91
2680 DATA A9,C8,C6,D0,F6,68,8D,30
2690 DATA C0,8A,18,6D,2E,C0,85,A7
2700 DATA 90,02,E6,A9,85,A7,A0,00
2710 DATA B1,A7,0D,05,C4,C0,B1,A7
2720 DATA 8D,06,C4,AC,12,C0,8D,30
2730 :
2740 DATA 0796
2750 :
2760 DATA C0,99,12,C0,4C,0C,C0,00
2770 DATA 48,00,48,A0,41,20,43,FF
2780 DATA 24,C4,A6,C4,D0,C4,C4,A5
2790 DATA C4,C4,DC,C4,00,00,00,00
2800 DATA D0,35,C4,37,C4,09,C4,20
2810 DATA C5,95,C5,11,C6,31,85,32
2820 DATA 20,20,20,20,20,00,00,00
2830 DATA 00,00,00,20,03,20,20,20
2840 :
2850 DATA 5630
2860 :
2870 DATA VEGE
```

Vízszintes ablak esetén megkérdezi a lehetőségek számát, különben azt az ablak magasságából automatikusan kiszámolja. Ha a lehetőségek megadásánál a módra és a címre is nullát írunk be, a cím értéke automatikusan 49164 (C00C) lesz, ahol egy RTS található.

Ha végeztünk az összes ablak megtervezésével, beírhatjuk őket a memóriába. Meg kell adni a kezdő tárolócímet, és a képernyőre kerülő ablakról eldönthetjük, hogy bekerüljön-e a memóriába vagy sem. Minden ablak tárolása után új címet adhatunk meg, vagy ha nem akarunk újat, az előzőleg eltárolt ablak után kerül az újabb ablak.

Ajánlatos nem számítgatni a következő ablak címét az ablakok tervezésénél, hanem miután már minden egyes ablak bent van a memóriában, egy monitorprogrammal írassuk be a szükséges címeket a megfelelő helyre.

Egy hasznos rutin

Az Ablakozónak az ablakokat a képernyőre másoló rutinjához közvetlenül is hozzáférhetünk. A memóriafelosztás a következő:

C009 — 49161 — a rutin hívási címe
C026 — 49190 — bal felső sarok x-koordinátája
C027 — 49191 — bal felső sarok y-koordinátája
C028 — 49192 — szélesség
C029 — 49193 — magasság
C02A — 49194 — tárcím (alsó bájt)
C02B — 49195 — tárcím (felső bájt)
C02C — 49196 — üzemmód

Ha az üzemmód 0, akkor az ablak a képernyőre töltődik, ha 1, akkor a memóriába. Az ablak mindig az utoljára elmentett ablak után kerül tárolásra. Betöltésnél, ha a tárcím felső bájtja nem 0, akkor a tárcím által megadott helyről töltődik. Ha a felső bájtja 0, akkor az utoljára tárolt ablak kerül a képernyőre. Ebben az esetben a C026—C02A bájtokat nem kell megadni. Minden egyes ablak elmentésénél növelni kell a már a lekérdezésnél említett szintmutató bájtot, ami a 49170-es címen található. Ennek elmulasztása esetén az előző ablak felülíródik. Képernyőre kerülésnél a bájt átírásával tetszőleges ablakot hozhatunk a képernyőre, de azért vigyázzunk, mert könnyen összekeveredik minden! *(3. program)*

S lőn BASIC

Végül két rövid rutin, amit felhasználtam a programok „BASIC-esítéséhez". Az egyik a tárban lévő bájtokból DATA sorokat készít, a másik ennek az ellenkezőjét végzi.

Írjuk be a kezdési címet a *4. program* 10-es sorába, majd indítsuk el. Ha végére értünk a DATA bájtoknak, állítsuk meg a programot RUN STOP-pal, majd töröljük 10-től 200-ig a sorokat, és mentsük el. Töltsük be az *5. programot,* töröljük a képernyőt, listázzuk ki az 5. programot és úgy töltsük be az előzőleg eltárolt DATA sorokat egyszerű LOAD-dal, hogy ez a program a képernyőn maradjon. Ezek után a képernyőn lévő minden egyes sorra rálépve nyomjunk RETURN-t.

Az első sorba beírva a betöltési kezdőcímet, kész az indításra váró programunk. Futtatás előtt azért mentsük el, sose lehet tudni, mit fog átírni. A rutin kis szépséghibája, hogy csak 64 bájtos blokkokban tud DATA tömböket előállítani.

Memóriaáttekintő

C000 — (49152) — Ablakozó rutin
C003 — (49155) — vissza az utolsó ablakba
C006 — (49158) — utolsó ablak törlése
C009 — (49161) — ablaktöltő, -mentő rutin
C00C — (49164) — RTS
C00D—C00E — (49165—49166) — az első ablak címe, alaphelyzetben $4000—16384
C012 — (49170) — szintjelző bájt
C013—C025 — (49172—49189) — az egyes szintekről hol lépett ki
C026—C02C — (49190—49196) — az ablaktöltő rutinnak az ablakot leíró bájtjai

Gárdonyi Gergely

1. program

```
1 REM ** 1-ES PELDAPROGRAM **
10 READ A:IF A=1000 THEN 50
20 POKE 16384+C,A:C=C+1
30 GOTO 10
50 PRINT"■":SYS49152
60 IF PEEK(49170)=1 THEN END
70 IF PEEK(49170)=0 THEN 90
80 GET A#:IF A#="" THEN 80
90 IF A#="I" THEN END
100 IF A#<>"N" THEN 80
110 SYS 49158
120 SYS 49155
130 GOTO 60
200 DATA 3,3,9,4,1,2,0,1,12,192,48,64
210 DATA 79,11,9,12,5,16,5,19,80
220 DATA 101,1,26,15,14,14,1,12,103
230 DATA 101,11,15,26,22,5,20,46,103,76
240 DATA 111,111,111,111,111,111,111,122
250 DATA 10,8,9,3,1,1,0,12,192
260 DATA 79,119,119,119,119,119,119,119
270 DATA 80,101,2,9,26,20,15,19,63,103
280 DATA 76,111,111,40,9,47,14,41,122
290 DATA 1000
```

2. program

```
1 REM ** 2-ES PELDAPROGRAM **
10 READ A:IF A=1000 THEN 50
20 POKE 16384+C,A:C=C+1
30 GOTO 10
50 PRINT"■":SYS49152
60 A=PEEK(49170)
70 IF A=0 THEN 100
80 IF PEEK(4917)>=0 THEN 120
90 PRINT "■1. ABLAK 2-ES":END
100 PRINT "■NEM VALASZTOTT SEMMIT!"
110 END
120 IF PEEK(49172)=0 THEN 140
130 IF PEEK(49172)=1 THEN 150
140 PRINT "■2. ABLAK 1-ES":END
150 IF PEEK(49173)=0 THEN 180
160 IF PEEK(49173)=1 THEN 190
170 IF PEEK(49173)=2 THEN 200
180 PRINT "■3. ABLAK 1-ES":END
190 PRINT "■3. ABLAK 2-ES":END
200 PRINT "■3. ABLAK 3-AS":END
210 DATA 5,5,3,4,1,2,1,0,24,64,12,192
220 DATA 79,119,80,101,1,103
230 DATA 101,2,103,76,111,122
240 DATA 8,10,3,4,1,2,0,1,12,192,48,64
250 DATA 79,119,80,101,1,103
260 DATA 101,2,103,76,111,122
270 DATA 13,0,3,5,1,3,0,0,0
275 DATA 12,192,12,192,12,192
280 DATA 79,119,80,101,1,103
290 DATA 101,2,103,101,3,103
295 DATA 76,111,122
297 DATA 1000
300 REM AZ ABLAKOK FELEPITESE
310 REM    1.      2.      3.
320 REM  ------------------------
330 REM    1---->  1-END    1-END
340 REM    2-END   2---->   2-END
350 REM                     3-END
```

3. program

```
1 REM ** 3-AS PELDAPROGRAM **
5 REM ** EGY 20*20-AS KEPERNYODARAB **
6 REM **          ELMENTESE          **
10 POKE 49190,5:POKE 49191,2
20 POKE 49192,20:POKE 49193,20
30 POKE 49196,1:POKE 49170,1
40 SYS 49161
50 REM ** EGY 20*20-AS KEPERNYODARAB **
60 REM **     BEHOZASA E000-ROL      **
70 POKE 49194,0:POKE 49195,240
80 POKE 49196,0
90 SYS 49161:GOSUB 300
100 REM * EGY 10*10-ES KEPERNYODARAB *
110 REM *          ELMENTESE          *
120 POKE 49190,8:POKE 49191,8
130 POKE 49192,10:POKE 49193,10
140 POKE 49196,1:POKE 49170,2
150 SYS 49161
160 REM * AZ ELSONEK ELMENTETT ABLAK *
170 REM * KEPERNYORE MASOLASA       *
180 POKE 49195,0
190 POKE 49196,0:POKE 49170,1
200 SYS 49161:GOSUB 300
210 REM * A MASODIKNAK ELMENTETT AB- *
220 REM * LAK KEPERNYORE MASOLASA   *
230 POKE 49195,0
240 POKE 49196,0:POKE 49170,2
250 SYS 49161
260 END
300 GET A#:IF A#="" THEN 300
310 RETURN
```

4. program

```
1 REM ** 4. PROGRAM - DATA-SITO **
10 S=1000:BA=49152
20 A#="0123456789ABCDEF"
30 PRINT"■":FOR F=0 TO 7:B#=""
40 FOR G=0 TO 7:A=PEEK(BA+G+F*8)
50 A1=INT(A/16)+1:A2=A-16*A1+17
60 B#=B#+MID$(A#,A1,1)+MID$(A#,A2,1)
70 B#=B#+CHR$(-44*(G<7))
80 C=C+A:NEXTG
90 PRINT S;"DATA ";B#:S=S+10:NEXT
100 PRINT "POKE 198,5:POKE631,13:";
110 PRINT "POKE632,13:POKE633,13:";
120 PRINT "POKE634,13:POKE635,13:"
130 PRINT "■■";S;":";S=S+10
140 PRINT S;"DATA";C:S=S+10
150 PRINT S;":"
160 PRINT "S=";S+10,":BA=";BA+64
170 PRINT "■■GOTO 20■"
180 POKE 198,10
190 FOR F=0 TO 9:POKE 631+F,13:NEXT
200 END
```

5. program

```
1 REM ** 5. PROGRAM - BETOLTO **
10 BA=49152
20 FOR F=0 TO 7:FOR G=0 TO 7
30 READ A#:IF A#="VEGE" THEN END
40 A1=LEFT$(A#,1):A2$=RIGHT$(A#,1)
50 A1=ASC(A1$)-48:A2=ASC(A2$)-48
60 A1=A1+7*(A1>9):A2=A2+7*(A2>9)
70 POKE BA+G+F*8,A1*16+A2
80 C=C+A1*16+A2:NEXT G,F
90 READ A:IF A<>C THEN PRINT "HIBA"
100 C=0:BA=BA+64:PRINT"■":BA:GOTO 20
```

Programozási fogások és melléfogások

Legutóbb egy plágium-esetről írtam. A „szerző" egy külföldi lapból kiollózott programot a saját neve alatt küldött be, ám balszerencséjére az eredeti program egy szépséghibáját is átvette, ami lehetővé tette az azonosítást, illetve a plágium felismerését. Az idegen tollakkal ékeskedés — amely anyagi előnyökkel is jár — sajnos egyáltalán nem ritka jelenség, bár sok esetben nem ilyen egyértelmű. Gyakran nehéz eldönteni, hogy véletlen egybeesésről, közkinccsé vált algoritmusok használatáról vagy tudatos „koppintásról" van-e szó. E sorozatban nem kívánok ezzel a témával foglalkozni, de előfordulhat, hogy közvetett módon szó esik róla.

Az *1. listán* látható rövid program a *BIT-LET* 1985. áprilisi számában jelent meg, szerzője a közismerten kis (3,5 kbájt) tárkapacitású VC20 gépi kódú programozásához kívánt vele segítséget nyújtani. Úgy látszik, a programban rejtőző csapdára senki sem hívta fel az olvasói észrevételekre mindig reagáló szerkesztőség figyelmét, így a programot tartalmazó teljes anyag változatlan formában bekerült az 1986 tavaszán megjelent *SZUPER BIT-LET*-be is. A kísérő szövegben a következőket olvashatjuk:

„Elterjedt programozási forma, hogy a gépi kódú anyagunkat egy BASIC sorban a REM utasításban tároljuk, kikeressük, hogy melyik memóriacímre helyezi az INTERPRETER, és onnan hívjuk meg gépi kódú szubrutinunkat. Másik forma, hogy DATA-ban tároljuk, és programfutáskor kiküldjük az adatokat egy letiltott területre, ahol az INTERPRETER nem nyúl hozzá. (...) A második formátum hátránya pedig a nagy helyfoglalás. Egyszer a DATA sorainkk foglalnak le a memóriában területet, másodszor pedig le kell tiltanunk egyéb szabad területet is, ahová küldjük a gépi kódú anyagunkat programfuttatáskor. A VC20-nál mindkét formátum adott. Azonban én ajánlok két *egyszerűbb és könnyen kezelhető programot*, amellyel gépi kódú anyagunkat kazettára vihetjük, és onnan tetszőleges helyre beírhatjuk."

Ezután a programlista, majd az idézett program részletes kezelési útmutatója következik, végül — szintén részletesen dokumentálva — a második program, amely az előzővel egy kazettás soros adatfájlba írt programot visszaolvassa, és a memória kívánt helyére tölti.

Úgy vélem, a program nem kíván magyarázatot. A gépi kódokat decimális adatok formájában a billentyűzetről kell beadni, és közvetlenül a mágnesszalagos adatfájlba kerülnek. Elég egyetlen apró hibát elkövetni, és máris kezdhetjük elölről a munkát. Egyszerű a megoldás: a kódokat mégis DATA utasításokba kell írni, és INPUT helyett READ-del olvasni, minden más változatlan marad. Hibázás esetén könnyen javíthatunk, de a végleges program futtatásakor — az eredeti elképzelésnek megfelelően — a DATA-k már nem foglalnak el helyet.

A VC20 fénykorában ismeretes volt egy jobb megoldás is, amellyel annak idején külföldi lapok programlistáin lépten-nyomon találkoztam. Egy úgynevezett előprogram (németül Vorprogramm) egymagában oldja meg a *BIT-LET*-ben közölt két kis program feladatát: ha kell, lefoglalja a szükséges helyet, majd DATA-kból betölti a gépi kódú adatokat, aztán (a mór megtette kötelességét...) törli magát a memóriából. Néha az előprogram másolta a karaktergenerátor ROM tartalmát a RAM-ba, máskor a főprogram betöltését is előkészítette.

Jobb módszerek is elterjedtek azóta, a VC20-as is kiment a divatból, de az eset mai szemmel nézve is tanulságos.

Az eredeti listához képest annyit módosítottam, hogy a sorok végéről elhagytam a REM utasításokat.

Általában morogni szoktam, amikor azt látom, hogy *kifejezetten C16-ra, illetve Plus/4-re írt* programban az egyszerű GETKEY utasítás helyett a VC20-on és C64-en megszokott — és ott indokoltan használt — programszerkezetet alkalmazzák. A számos példa közül a *Mikrovilág* 1989/1. számában közölt *Grafikaözön* című programból emeltem ki egy kétsoros részletet, amelyet a *2. listán* mutatok be. A program egy olyan sorozatnak része, amely éppen a C16 és a Plus/4 speciális grafikai lehetőségeit mutatja be. Az egyes bemutatók után tetszőleges billentyű leütésével lehet tovább lépni. Erre a program hét pontjáról hívják meg a 370-es sorban lévő szubrutint. Nyilvánvaló, hogy egyszerűbb lenne minden GOSUB 370 helyére GETKEY A$-t írni. Csak ráadás, hogy a szubrutin előtt hiányzik az END, így a program egy „RETURN WITHOUT GOSUB" hibaüzenettel fejezi be futását. Szerencsére akkor, amikor különben is befejezné.

Ez év tavaszán az akkor hetedik általánosba járó Wolf Erik hívta fel a figyelmemet arra, hogy a *PLUS/4-es Felhasználói kézikönyv* (Novotrade, 1987.) 76. oldalán közölt *A PLUS/4-es mint „zenegép"* program szintaktikai hiba miatt nem működik. Együtt böngészve a programot tisztáztuk, hogy a nyomdai szedéssel készült lista 45-ös és 47-es sorának végéről hiányzik egy-egy befejező zárójel. (Pontosabban: a sorban lévő utolsó vessző előtt.) A nagy meglepetés — mivel eredeti Commodore-kiadvány fordításáról van szó — a program 30-as sora volt, amelyet a *3. listán* mutatok meg. Ismétlem: *nem fordítási hiba, az eredetiben is ez áll.* Ha a Commodore cég a *GETKEY utasítást is ismertető Felhasználói kézikönyvében* ilyen megoldást javasol, akkor mit várhatunk más programozóktól? Talán olyan eredeti megoldásokat, amilyet a *Mikrovilág* 1989/16. számában közölt *Egyszeregy* című programban találtam, és a *4. listán* mutatok be. Lehet ezt még tovább fokozni?

Barna László

1. lista

```
1000 OPEN1,1,1
1001  INPUTA:IFA<0THEN1005
1002 K=K+A:N=N+1
1003 PRINT#1,A
1004 GOTO1001
1005 CLOSE1:PRINTK;N
1006 NEW
```

2. lista

```
    ...
360 GOSUB 370
370 GET A$:IF A$="" THEN 370:ELSE RETURN
    ...
```

3. lista

```
    ...
30 GET A$:IF A$="" THEN 30
    ...
```

4. lista

```
    ...
60  GET KEY C$:IF C$="" THEN 60:ELSE 70
70 ...
```

Csatlakozzunk, de mivel?

Már a C16-os és a C Plus/4-es gépek megjelenésekor a vevők bizonyos fokú zsarolásnak tekintették a gyártók részéről azt a tényt, hogy a botkormány csatlakozási felülete nem szabványos, és speciális botkormánnyal vagy ún. joystick-adapterrel lehet csak a számítógéphez botkormányt csatlakoztatni. De amikor az első Enterprise gépek megjelentek a Centrum Áruházak műszaki osztályain, sokan úgy vélték, hogy ez a gép éppen azért, mert ún. NYÁK-kivezetéses csatlakozási felülete van (csak az RF antenna csatlakozási pontja szabványos), hamarosan meg fog bukni a hazai számítástechnikai piacon.

Való igaz, közel két évnek kellett eltelnie ahhoz, hogy ma már minden egyes csatlakozási ponthoz akár OEM, akár kész állapotban lévő kábeleket, adaptereket lehessen kapni a géphez! A fenti NYÁK-kiosztásos filozófiát számos világhírű cég tette sajátjává, például a Sony, a Sharp, az Amstrad, a Philips és a Scheider.

Ma már úgy ítélhetjük meg, hogy a NYÁK-lap, illetve az alaplapok ilyen megtervezése és legyártatása még az aranyozott csatlakozási pontok ellenére is jóval olcsóbb, mint ha valamennyi kimeneti pontot szabványos NYÁK-ba ültethető csatlakozóval látták volna el. Azaz az egységnyi számítógépben megtestesülő kivezetési pontok előállítási költsége lényegesen olcsóbbá vált.

A koncepció áldozatai

A gyártók már jóval kevesebbet törődtek a járulékos beruházásnak számító interfészek és speciális kábelek előállításával, hiszen a fogadó oldaláról kellett biztosítani azt a csatlakozási felületet, amelyet több esetben éppen a számítógépből spóroltak ki. Nos, ennek a hibás fejlesztési koncepciónak esett áldozatul maga az Enterprise is. Igaz, hogy ma már mindenféle csatlakozókábel és interfész kapható az EP-hez, de ha összeadjuk ezek értékét, meghaladja a 4000 forintot. Bár az is tény, hogy nem kell azt sem egyszerre megvenni!

Az Enterprise valamennyi csatlakozókábelét és magát a botkormány-interfészt is a Sprint Kisszövetkezet gyártotta. Az összes csatlakozási felület más-más kiosztású, és még véletlenül sem lehet őket felcserélni. Ezért nem tipizálhatók, és mindegyik kábel elkészítéséhez egy külön szerszámot kellett gyártani. A kisszövetkezet a Kontakta gyár csatlakozóit használta fel, az esetek többségében. A manuális szerelést követően úgy összeragasztották a csatlakozófejeket, hogy azokat egy esetleges javításkor nem lehet szétszedni.

A kábelek választékában csupán a Centronics printerkábel és az EURO-SCART-os monitorkábel származik tőkés importból. A forgalmazó által jelzett adatok szerint eddig csak a botkormány-adapterek szerelése során jelentkezett hibás termék. A kábel, illetve interfész bedugását a számítógép hátsó kimeneti pontjába megkönnyíti a csatlakoztatandó kiegészítőnek egy hornya (vájata), melynek ellenpólusa éppen a számítógép csatlakozási felületével esik egybe.

A forgalmazás kezdetén előfordultak olyan idegesítő hibák is, hogy a „mother board" 4 pontos felcsavarozása 2—3 milliméterrel balra vagy jobbra sikerült a NYÁK-kivezető sínek eredetileg tervezett és a műanyag házzal összemért elhelyezkedésétől. Ennek az lett az eredménye, hogy bár a csatlakozókábel-doboz a vezető sínre rácsúszott, a NYÁK kivezetési pontjai sohasem érintkezhettek a kábel csatlakozási pontjaival. Az EP Márkaszerviz azonban ezt a problémát is megoldotta: az alaplap tartó furatait kiszélesítette, és az alaplapot balra vagy jobbra elmozgatta olyan mértékben, hogy most már a csatlakoztatandó kábel felfeküdhessen a NYÁK-lapon.

Tv és számítógép kapcsolata

Az otthoni számítógépek legtöbbjéhez a tv-készüléket használják monitorként. A tv-hez a számítógépet háromféleképpen lehet illeszteni:

— RF, rádiófrekvenciásan (97%),

— CVS, azaz Composit Video Signal segítségével (2,6%),

— RGB, azaz a három színcsatornát és a szinkroncsatornát külön-külön (0,4%).

Zárójelben az EP-tulajdonosok alkalmazási mutatóit tüntettük fel.

Mielőtt végignéznénk a lehetőségek közötti különbségeket, beszéljünk pár szót arról is, hogy hogyan jelenik meg a tv-kép az adókból sugárzott jel hatására. A tv-adó ún. RF jeleket sugároz, ezeket érzékeli az antenna, és egy antennakábelen keresztül továbbítja a tv-készülékbe. Itt először egy olyan áramkörbe kerül általában, amely előállítja az ún. Composit Video Signalt (CVS-t). A CVS egy másik áramkörbe jut, ahol különválasztódnak a színjelek — piros, zöld és kék —, valamint a megjelenítéshez szükséges szinkronjel (RGBY). A színjelek egy illesztő fokozat után közvetlenül a képcső színágyúját vezérlik. A szinkronjelet egy újabb áramkör választja szét a képcső vezérléséhez szükséges vezérlőjelekre, amelyek a kép megjelenítését irányítják. Ez a leírás azonban a végletekig leegyszerűsített változata annak, ahogyan az adóból érkező jelből kép lesz. Tudni kell azt is, hogy minden jel, amely áthalad egy áramkörön, nemcsak átalakítást, illesztést végez, hanem „zajt" is ad hozzá a jelhez. Természetesen ez a „zaj" a jelhez képest igen kicsi, de minél több áramkörön halad át a jel, annál több zaj kerül bele, és ez az összegeződés a végén már zavaró lehet.

RF összeköttetés esetén a jel igen zajos, hiszen az összes áramkörön áthalad. Maga az RF-CVS, illetve a CVS-RF eszköz a legzajosabb, tehát a legnagyobb nyereséget úgy tudjuk csak elérni, ha ezt az átalakítást és visszaalakítást kiiktatjuk.

Composit Video Signal összeköttetés esetén az előző összeköttetéshez képest már igen látványos javulás várható:

— megszűnik a nagy egyszínű foltok „grízes" vibrálása,

— láthatóvá válik a sárga alapon a fehér pont is,

— olvashatóak lesznek a betűk a 80 karakteres üzemmódban,

— megszűnik a kép „hullámzása", remegése,

— az eddig összemosódott foltok jól látható határvonalat kapnak,

— az RF hurok okozta programbetöltési gondok teljesen megszűnnek, nem kell az egyes programok esetén a kábelt ki-be húzogatni.

Az újabb típusú Orion és Videoton tv-k DIN, a nyugati tv-k Siemens, ITT EUROSCART bemenettel rendelkeznek.

RGBY összeköttetésnél legrövidebb a jel útja: RGBY → kábel → RGBY. Az előzőekhez képest a javulás még szembetűnőbb: a kép tisztább, a színek kontrasztosak, a változás azonban már nem olyan mértékű, mint az RF és a CVS közötti volt. A normál tv-k nagy többsége nem rendelkezik ilyen csatlakozási lehetőségekkel. Sajnos a személyi számítógépeknél sincs meg mindenhol ez a kimeneti lehetőség. Az EP-nél, az Amstrad és a Scheider számítógépeknél azonban megtalálható. Az Enterprise által biztosítható Interlace technika már csak ilyen nagy felbontású monitorral érhető el. Ilyen esetben a pixelek száma a képernyőn eléri az 1280×720-as képfelbontást.

Hasznos tanácsok

Ha a tv-n vagy monitoron a számítógép képét nézzük, vegyük a megszokottnál kisebbre a fényerőt, a kontrasztot és a színtelítettséget. Lehetőleg ügyeljünk arra, hogy ha nem szükséges, ne legyenek a képen sokáig egy helyben maradó fények, jelek (betűk), mert azok a képben „beégést" okozhatnak.

Használjunk antisztatikus tisztítófolyadékot a képernyő tisztítására!

Monitorkábelek Enterprise gépekhez

1. BNC-mono/sztereo 1998,— Ft
   - SONY
   - JVC
2. RCA-Pin JACK 1298,— Ft
   - Philips
   - Commodore
   - Samsung
   - NEC
3. EUROSCART 1298,— Ft
   - ITT
   - Siemens
   - Thomson
   - Grundig
   - Waltham
   - Loewe
4. 6 pólusú DIN 1298,— Ft
   - Orion tv
   - Videoton tv
5. OEM nyílt végű monitorkábel 1148,— Ft
6. TV-RF antennajel-erősítő doboz
   - Beijing (kínai) tv
   - Videoton
   - Orion

Minősítés

Annak ellenére, hogy a monitorkábelek árai különböző szolgáltatásokat is magukba foglalnak — felvilágosítás, szaktanácsadás, az Enterprise számítógép átalakítása az RGBY fogadására garanciával —, meg kell állapítani, hogy az OEM termékek kivételével a csatlakozókábelek drágák. A műanyag palástok összeragasztása során előfordul, hogy a ragasztó kifolyik, és a termék esztétikai értékét erősen rontja. A termék kiszerelése sem áll arányban a magas árral.

Vizsgáljuk meg a kábelek, illetve interfészek árait magának a számítógépnek az árához viszonyítva. Az Enterprise 128K számítógép ára 9650 forint.

Botkormány-adapter:	456 Ft	0,5%
EUROSCART monitorkábel:	1298 Ft	13,5%
Centronics printerkábel:	1498 Ft	15,5%
Serial NET dual kábel:	898 Ft	0,9%

Az EUROSCART és az RF csatlakozók használatán kívül, bármilyen más monitorkábel alkalmazása esetén az Enterprise márkaszervizekhez kell fordulni az átalakítások garanciális elvégzéséért. Úgy ítéljük meg, hogy csak abban az esetben csökkentjük kiadásainkat, ha nyílt végű OEM kábeleket vásárolunk. Ez azonban feltételezi, hogy a műszaki életben jártasak vagyunk, és igen jól ismerjük a NYÁK-lap kivezetési pontjainak definiált funkcióit, nem beszélve arról a tényről, hogy a kábel másik oldalán lévő csatlakozóval már rendelkezünk!

A forgalmazó egyértelmű hibájának róható fel, hogy mind a mai napig nem hajlandó félkész termék formájában műanyag palástokat forgalomba hozni azok számára, akik maguk akarnák a fenti termékeket összeszerelni.

Mindent egybevetve azonban, dicséretes gyorsasággal fejlesztette ki a forgalmazó a különféle kábeleket és interfészeket az Enterprise számítógéphez. Ezt a kényszerű fejlesztést még ha a forgalmazó nem is akarta, akkor is meg kellett volna tennie, mivel a mai napig ilyen széles választék a Nyugaton forgalomba hozott Enterprise számítógépekhez csak hazánkban található.

Vajha egyszer elérhetnénk, hogy mi exportálhatunk számítástechnikai tartozékokat, kiegészítőket Angliába, Hollandiába és az NSZK-ba!

Kérdések

Szokásunkhoz híven most is néhány olyan kérdéssel zárjuk az elemzést, amelyekre mind a mai napig nem ismerjük a választ! Miért nincs válasz arra az „RF-hurok" problémára, amelynek sajnálatos következményeképpen néhány kazetta betöltésekor ki kell húzni az RF-kábelt a tv-ből (például Nautilus)? Lehet, hogy az „RF-hurok" az Enterprise Bermuda-háromszöge? Nem lett volna egyszerűbb valamennyi forgalomba kerülő számítógéppel együtt magához a géphez adni — vagy talán a számítógép dobozában elhelyezni — az oly nélkülözhetetlen botkormány-adaptert? Miért olyan kevés az a játék, amellyel párosban is lehet játszani?

P. T.

Minden kedden 17-től 20 óráig
HCC ENTERPRISE klub
a VSZM
Közösségi Házban
(Bp. XI., Fehérvári út 120.)
Klubvezető: Romvári Gábor
Telefon: 181-0950/473

Olvasóink közül néhányan bizonyára az anyagnyilvántartó programok iránt is érdeklődnek. Úgy érezzük, érdemes egyszer egy ilyen programot is közölnünk, mivel az adatmozgatási lehetőségekről eddig nem sokat írtunk. Ezen túlmenően ez a program nemcsak az anyagok nyilvántartására alkalmas, hanem kis átalakítással háztartásokban is használhatjuk. Felhívjuk az olvasók figyelmét a „név szerinti keresés" opció szellemes megoldására. Ugyanis nincs szükség a keresett tárgy teljes nevének begépelésére, elég, ha annak kezdő-, illetve néhány első betűjét ismerjük.

A gőgös spectrumos esete az Enterprise-zal

Spectrumon nevelkedett BASIC-rajongók, akik más számítógépek BASIC-jébe is bele-bele kóstolgatnak, gyakran úgy érzik, hogy a másik gépen semmiség eljutni arra a programozási szintre, amit a jó öreg Spectrumon tudtak, tudnak.

Az Enterprise-BASIC sok hasonlóságot mutat a Spectrumével, bár annál lényegesen több szolgáltatást nyújt (például AUTO, RENUMBER, DELETE, DO-LOOP).

Egyik ismerősömnek kicsit gőgösen jelentettem ki, nem is olyan régen, hogy nem lehet bonyolult dolog az Enterprise gépen egy anyagnyilvántartó programot megírni BASIC-ben.

Ezt a kijelentésemet az Enterprise ismerete nélkül tettem. Egy esős délután neki is láttam, hogy nagyképű kijelentésemet mintegy aláhúzandó, megírjak részére egy menüvezérelt nyilvántartó programot. A program ma már működik, de azóta én is sokkal szerényebb vagyok.

A program lényege, hogy a képernyőn megjelenő menüből az ember kiválasztja a neki megfelelő opciót, és ha ott már kiélte magát, visszatérhet a menühöz, hogy újabbat válasszon.
A bejelentkező képernyőn az alábbi látható:

KÉSZLET-PM
1. BEÍRÁS
2. NÉVMÓDOSÍTÁS
3. KERESÉS NÉV SZERINT
4. LISTÁZÁS
5. KIADÁS
6. BEVÉTEL
7. ADATMENTÉS
8. ADATTÖLTÉS

A program megírása az első hat pontig tulajdonképpen nem izzasztott meg, bár el kell mondanom, hogy szokatlan volt mindvégig az, hogy az utasításokat, függvényeket karakterenként kellett begépelni. Furcsa volt az is, hogy nagyméretű tömböt nem tudtam létrehozni a nekem megszokott DIM utasítással.

Az addig kézbe sem vett felhasználói kézikönyv azonban segített, s így akadtam rá a NUMERIC és STRING tömbdeklarációra. A lényeg az, hogy nem túl nagy küzdelem árán megírtam egy olyan programot, amely 1000 adat beírását, lekérdezését, módosítását stb. teszi lehetővé.

A 7. és 8. pontok megírását a végére hagytam, „tudván", hogy ez a legegyszerűbb dolog a világon. Próbaképpen kiadtam az utasítást:

SAVE „ADAT" DATA A ( )

Megnyugodva hallottam, hogy a gép sípol, nyöszörög. Hamarosan lelohadt a kedvem, mikor tapasztaltam, hogy az általam hitt adatok helyett a BASIC programot menti. Lázasan lapozgatni kezdtem a kézikönyvet, ami a SAVE utasítást taglaló részben megnyugtatott, hogy a SAVE itt is SAVE, valamint bátorítást adott, hogy a fájl itt is lehet adat- és programfájl.

Sajnos e témában többet nem találtam, már úgy értem, olyat, ami nekem tetszett. Megpróbáltam mindent:

```
SAVE „ADAT" FILE a ( )          ! EREDMÉNYTELEN
SAVE „ADAT" NUMERIC a ( ) ! EREDMÉNYTELEN
STB                             ! EREDMÉNYTELEN
```

Hosszas telefonálgatásba kezdtem, melynek végén megtudtam, hogy figyelmesebben kell áttanulmányozni a kézikönyvet, mert ott rá lehet találni az OPEN utasításra. Elszégyellve magam, megköszöntem a jó tanácsot, és megígértem, hogy az eredményről beszámolok.

Nem fárasztom az olvasót azzal, hogy az összes botlásomat kifejtsem, inkább leírom az általam megtalált — és hangsúlyozom — működő megoldást, tudva azt is, hogy bizonyosan van elegánsabb, profibb megközelítés is.

Az *1. lista* tartalmazza az anyagnyilvántartó programot. Ez a változat kimenti az adatokat a kazettára. Az adatok mentése a 7000-as sortól — amikor megnyitjuk az adatcsatornát —, a beolvasás a 8000-es sortól kezdődik. A lemezmeghajtóra írt változat lényegében csak az adatok kimentésének és betöltésének módjában különbözik, ezért csak az erre vonatkozó programrészletet közlöm *(2. lista)*.

Borzasztó dolog, ha adatkezelő programunk hibaüzenettel áll le, mivel újraindításkor ez adatvesztést eredményez. Fokozottan áll fenn ez a veszély adatmentés és adattöltés során, ezért keretezi a mentő-töltő rutinokat egy-egy hibakezelő rutin. A hibakezelő rutin lényege, hogy bármilyen hiba esetén a program visszatér a főmenühöz. Így, ha véletlenül elindítanánk a töltést, illetve a mentést, a STOP billentyű megnyomása sem okoz leállást.

Meg kell jegyezni azt is, hogy 1000 adat magnóra mentése, ha a STRING tömb elemei 22 karakter hosszúak, mintegy 5-6 percet vesz igénybe. Mágneslemezre mentéskor ez az idő kb. 30 másodperc. A Spectrum-tulajdonosok számára furcsa lehet az is, hogy a gép szakaszosan ment, „gondolkodik". A magyarázat az, hogy a gép a memóriáját 4 kbájtos szegmensekben menti kazettára. Érdekes viszont, hogy mágneslemez esetén a szakaszosan mentett adatokat megszakítások nélkül tölti viszsza.

Papp Miklós

1. lista

```
  1 GOTO 9000
 11 TEXT
 21 PRINT AT 2,18:"KESZLET"
 31 PRINT AT 5,13:"1. BEIRAS"
 41 PRINT AT 7,13:"2. NEV JAVITAS"
 51 PRINT AT 9,13:"3. KERESES NEV."
 61 PRINT AT 11,13:"4. LISTAZAS"
 71 PRINT AT 13,13:"5. KIADAS"
 81 PRINT AT 15,13:"6. BEVETEL"
 91 PRINT AT 17,13:"7. ADAT MENTES"
101 PRINT AT 19,13:"8. ADAT TOLTES"
111 LET B$=INKEY$
121 IF B$="1" THEN GOTO 1000
131 IF B$="2" THEN GOTO 2000
141 IF B$="3" THEN GOTO 3000
151 IF B$="4" THEN GOTO 4000
161 IF B$="5" THEN GOTO 5000
171 IF B$="6" THEN GOTO 6000
181 IF B$="7" THEN GOTO 7000
191 IF B$="8" THEN GOTO 8000
201 GOTO 111
211 STOP
1000 TEXT
1010 PRINT AT 5,18:"TETEL=";A(901)
1020 PRINT AT 19,0:"ARU KODJA ?"
1030 INPUT AT 20,0:C$
1040 IF LEN(C$)>6 THEN GOTO 1000
1050 IF LEN(C$)<1 THEN GOTO 11
1060 PRINT AT 19,0:"ARU MEGNEVEZESE"
1070 PRINT AT 20,0:"
                    "
1080 INPUT AT 20,0:B$
1090 IF B$="" THEN GOTO 11
1100 IF LEN(B$)>22 THEN GOTO 1000
1110 PRINT AT 19,0:"ARU MENNYISEGE "
1120 PRINT AT 20,0:"
                    "
1130 INPUT AT 20,0:D$
1140 LET S=0
1150 PRINT AT 10,4:C$;" ";B$
1160 FOR G=1 TO LEN(D$)
1170    IF ORD(D$(G))>57 THEN LET S=1
1180 NEXT G
1190 IF S=1 THEN GOTO 1120
1200 LET B=VAL(D$)
1210 PRINT AT 12,10:"MENNYISEGE";B
1220 PRINT AT 5,18:"TETEL=";A(901)
1230 PRINT AT 20,0:"
                    "
1240 PRINT AT 19,0:"
                    "
1250 PRINT AT 19,0:"MEGFELEL i/n"
1260 LET G$=INKEY$
1270 IF G$="" THEN GOTO 1250
1280 IF G$="n" THEN GOTO 1000
1290 LET A(A(901))=B
1300 LET A$(A(901))=B$
1310 LET S$(A(901))=C$
1320 LET A(901)=A(901)+1
1330 GOTO 1000
2000 TEXT
2010 PRINT AT 20,0:"IRJA BE AZ ARU KODO
T"
2020 INPUT AT 21,0:B$
2030 IF LEN(B$)>6 THEN GOTO 2000
2040 IF B$="" THEN GOTO 11
2050 LET D=LEN(B$)
2060 TEXT
2070 LET I=1
2080 LET C$=S$(I)
2090 IF B$=C$ THEN GOTO 2130
2100 LET I=I+1
2110 IF I>A(901) THEN GOTO 2000
2120 GOTO 2080
2130 TEXT
2140 PRINT AT 10,4:S$(I);" ";A$(I)
2150 PRINT AT 20,0:"UJ MEGNEVEZES ?"
2160 PRINT AT 21,0:" "
2170 INPUT AT 21,0:D$
2180 IF LEN(D$)>22 THEN GOTO 2130
2190 PRINT AT 12,10:"UJ MEGNEVEZES"
2200 PRINT AT 14,4:S$(I);" ";D$
2210 PRINT AT 20,0:"MEFELEL I/N
                    "
2220 PRINT AT 21,0:"
                    "
2230 LET B$=INKEY$
2240 IF B$="" THEN GOTO 2230
2250 IF B$="i" THEN LET A$(I)=D$:GOTO 1
1
2260 IF B$="n" THEN GOTO 2130
2270 GOTO 2230
3000 TEXT
3010 PRINT AT 20,0:"IRJA BE AZ ARU NEVE
T:"
3020 INPUT AT 21,0:B$
3030 IF LEN(B$)>22 THEN GOTO 3000
3040 IF B$="" THEN GOTO 11
3050 LET D=LEN(B$)
3060 TEXT :LET C=1
3070 FOR I=1 TO A(901)
3080    LET C$=A$(I)
3090    IF B$=C$(1:D) THEN LET C=C+1:PRI
NT A$(I):PRINT "MENNYISEG=";A(I):PRINT "
KODSZAM=";S$(I):PRINT "==================
======================"
3100    IF C>5 AND INKEY$="" THEN PRINT
AT 22,0:"TOVABB=ENTER":GOTO 3100
3110    IF C>5 THEN TEXT :LET C=1
3120 NEXT I
3130 PRINT "**************************
***********"
3140 PRINT AT 21,0:"TOVABB i/n"
3150 LET B$=INKEY$
3160 IF B$="" THEN GOTO 3150
3170 IF B$="i" THEN GOTO 3000
3180 IF B$="n" THEN GOTO 11
3190 GOTO 3150
4000 TEXT :LET S=0
4010 PRINT AT 19,0:"HANYADIK TETELTOL ?
"
4020 INPUT AT 20,0:B$
4030 FOR D=1 TO LEN(B$)
4040    IF ORD(B$(D))>57 THEN LET S=1
4050 NEXT D
4060 IF S=1 THEN GOTO 4000
4070 IF VAL(B$)<1 THEN GOTO 4000
4080 IF VAL(B$)>A(901) THEN GOTO 4000
4090 LET I=VAL(B$)
4100 LET C=0
4110 TEXT
4120 PRINT A$(I)
4130 PRINT "MENNYISEG=";A(I)
4140 PRINT "KODSZAM=";S$(I)
4150 PRINT "--------------------------
-----------"
4160 LET I=I+1
4170 LET C=C+1
4180 IF I>=A(901) THEN GOTO 4210
4190 IF C>=4 THEN GOTO 4270
4200 GOTO 4120
4210 PRINT AT 21,0:"UJRA i/n"
4220 LET B$=INKEY$
4230 IF B$="i" THEN GOTO 4000
4240 IF B$="n" THEN GOTO 11
4250 IF B$="" THEN GOTO 4220
4260 GOTO 4210
4270 PRINT AT 21,0:"TOVABB i/n"
4280 LET B$=INKEY$
4290 IF B$="i" THEN GOTO 4100
4300 IF B$="n" THEN GOTO 11
4310 IF B$="" THEN GOTO 4280
4320 GOTO 4270
5000 TEXT
5010 PRINT AT 20,0:"KODSZAM   ?"
5020 INPUT AT 21,0:B$
5030 IF B$="" THEN GOTO 11
5040 IF LEN(B$)>6 THEN GOTO 5000
5050 LET G=1
```

```
5060 IF B$=S$(G) THEN GOTO 5110
5070 LET G=G+1
5080 IF G>A(901) THEN GOTO 5000
5100 GOTO 5060
5110 PRINT AT 5,4:S$(G);" ";A$(G)
5120 PRINT AT 7,10:"MENNYISEG=";A(G)
5130 PRINT AT 20,0:"KIADOTT ?  "
5140 PRINT AT 21,0:"           "
5150 INPUT AT 21,0:C$
5160 LET S=0
5170 FOR I=1 TO LEN(C$)
5180   IF ORD(C$(I))>57 THEN LET S=1
5190 NEXT I
5200 IF VAL(C$)<1 THEN GOTO 5150
5210 IF S=1 THEN GOTO 5150
5220 LET C=VAL(C$)
5230 PRINT AT 9,10:"KIADOTT=";C
5240 PRINT AT 11,10:"TEHAT= ";A(G)-C
5250 PRINT AT 20,0:"MEGFELEL i/n"
5260 PRINT AT 21,0:"           "
5270 LET B$=INKEY$
5280 IF B$="" THEN GOTO 5270
5290 IF B$="n" THEN GOTO 5000
5300 IF B$="i" THEN LET A(G)=A(G)-C
5310 GOTO 5000
6000 TEXT
6010 PRINT AT 20,0:"KODSZAM   ?"
6020 INPUT AT 21,0:B$
6030 IF B$="" THEN GOTO 11
6040 IF LEN(B$)>6 THEN GOTO 6000
6050 LET G=1
6060 IF B$=S$(G) THEN GOTO 6110
6070 LET G=G+1
6080 IF G>A(901) THEN GOTO 6000
6100 GOTO 6060
6110 PRINT AT 5,4:S$(G);" ";A$(G)
6120 PRINT AT 7,10:"MENNYISEG=";A(G)
6130 PRINT AT 20,0:"BEVETEL ?  "
6140 PRINT AT 21,0:"           "
6150 INPUT AT 21,0:C$
6160 LET S=0
6170 FOR I=1 TO LEN(C$)
6180   IF ORD(C$(I))>57 THEN LET S=1
6190 NEXT I
6200 IF VAL(C$)<1 THEN GOTO 6150
6210 IF S=1 THEN GOTO 6150
6220 LET C=VAL(C$)
6230 PRINT AT 9,10:"BEVETEL=";C
6240 PRINT AT 11,10:"TEHAT= ";A(G)+C
6250 PRINT AT 20,0:"MEGFELEL i/n"
6260 PRINT AT 21,0:"           "
6270 LET B$=INKEY$
6280 IF B$="" THEN GOTO 6270
6290 IF B$="n" THEN GOTO 6000
6300 IF B$="i" THEN LET A(G)=A(G)-C
6310 GOTO 6000
7000 TEXT
7010 INPUT AT 22,0,PROMPT "ADAT NEVE=":
C$
7020 HANDLER PAPP
7030   IF EXTYPE>0 THEN GOTO 11
7040 END HANDLER
7050 WHEN EXCEPTION USE PAPP
7060   OPEN #106:"TAPE:"&C$ ACCESS OUTP
UT
7070   FOR I=1 TO 901
7080     PRINT #106:A(I)
7090   NEXT I
7100   FOR I=1 TO A(901)-1
7110     LPRINT #106:S$(I)
7120   NEXT I
7130   FOR I=1 TO A(901)-1
7140     LPRINT #106:A$(I)
7150   NEXT I
7160   CLOSE #106
7170 END WHEN
7180 GOTO 11
8000 TEXT
8010 HANDLER MIKLOS
8020   IF EXTYPE>0 THEN GOTO 11
8030 END HANDLER
8040 WHEN EXCEPTION USE MIKLOS
8050   OPEN #106:"TAPE:" ACCESS INPUT
8060   FOR I=1 TO 901
8070     INPUT #106:A(I)
8080   NEXT I
8091   FOR I=1 TO A(901)-1
8093     LINE INPUT #106:S$(I)
8094   NEXT I
8110   FOR I=1 TO A(901)-1
8120     LINE INPUT #106:A$(I)
8130   NEXT I
8140   CLOSE #106
8150 END WHEN
8160 GOTO 11
9000 NUMERIC A(901)
9010 LET A(901)=1
9020 STRING A$(1 TO 900)*22
9030 STRING S$(1 TO 900)*6
9040 FOR I=1 TO 900
9050   LET A$(I)="URES.................
"
9060   LET S$(I)="000000"
9070   LET A(I)=0
9080 NEXT I

9090 GOTO 11
```

2. lista

```
7000 TEXT
7010 INPUT AT 22,0,PROMPT "ADAT NEVE=":
C$
7020 HANDLER PAPP
7030   IF EXTYPE>0 THEN GOTO 11
7040 END HANDLER
7050 WHEN EXCEPTION USE PAPP
7060   OPEN #106:C$ ACCESS OUTPUT
7070   FOR I=1 TO 901
7080     PRINT #106:A(I)
7090   NEXT I
7100   FOR I=1 TO A(901)-1
7110     LPRINT #106:S$(I)
7120   NEXT I
7130   FOR I=1 TO A(901)-1
7140     LPRINT #106:A$(I)
7150   NEXT I
7160   CLOSE #106
7170 END WHEN
7180 GOTO 11
8000 TEXT
8005 INPUT AT 22,0,PROMPT "ADAT NEVE=":
C$
8010 HANDLER MIKLOS
8020   IF EXTYPE>0 THEN GOTO 11
8030 END HANDLER
8040 WHEN EXCEPTION USE MIKLOS
8050   OPEN #106:C$ ACCESS INPUT
8060   FOR I=1 TO 901
8070     INPUT #106:A(I)
8080   NEXT I
8091   FOR I=1 TO A(901)-1
8093     LINE INPUT #106:S$(I)
8094   NEXT I
8110   FOR I=1 TO A(901)-1
8120     LINE INPUT #106:A$(I)
8130   NEXT I
8140   CLOSE #106
8150 END WHEN
8160 GOTO 11
9000 NUMERIC A(901)
9010 LET A(901)=1
9020 STRING A$(1 TO 900)*22
9030 STRING S$(1 TO 900)*6
9040 FOR I=1 TO 900
9050   LET A$(I)="URES.................."
9060   LET S$(I)="000000"
9070   LET A(I)=0
9080 NEXT I
9090 GOTO 11
```

A LORIGRAPH RAJZOLÓPROGRAM HASZNOSÍTÁSA II.

A LORIGRAPH karaktertervező részével készített karakterek háttértárolóra menthetők. A program az így kimentett karakterek betöltését végzi, és átállítja a karakterkészletet a tervezettnek megfelelően. Ez akkor gyorsabb, ha sok karaktert kell átdefiniálni, és az BASIC módszerrel már lassan menne. A felhasználó is jobban elviseli a hosszabb töltési időt, mint ha percekig kell várnia, mire a program életjelt ad magáról.

A programban a beolvasó csatorna megnyitása és lezárása nem gépi kódban íródott, hanem BASIC-ből történik. Így könnyebb a fájl nevének megváltoztatása. A képbetöltő programnál is megváltoztathatjuk ilyenre a csatornamegnyitásokat. Abban a programban azért alkalmaztam a gépi kódú megnyitási formát, hogy megismerjék, de jó példának mutatkozott arra is, hogy bemutathassam, hogyan lehet több adatot átadni a gépi kódú rutinjainknak. A program átalakítása nem okozhat gondot, ha a leírást megfelelően tanulmányozzák.

Az *1. listán* látható program tölti be a karakterkészletet. A *2. lista* a gépi kódú rész assembly listája, ami tulajdonképpen megfelel a 40-es sor hexadecimális számsorozatának. A programot a 80-as sortól tovább írhatjuk, illetve a meglévő programjainkat MERGE-ölhetjük. Ha szalagos rendszerrel dolgozunk, akkor a karakterkészletet a kimentett program mögé vegyük fel.

A LORIGRAP program lemezre is felvihető. A LORIGRAP; LORI2; LORI3 fájlneveket használja, amiből kitűnik a probléma, hogy a harmadik fájl neve megegyezik a kimentendő rajz vagy karakterkészlet nevével. Kár, hogy a programozók nem gondoltak a lemezegységre — vagy esetleg ez bizonyítja, hogy igenis gondoltak?!

A megoldás az lehet, hogy betöltés után kicseréljük a programot tartalmazó lemezt egy adatlemezre. Ha ezt elfelejtjük, kimentéskor felülírjuk a program harmadik fájlját, s ezzel tönkretesszük a LORIGRAPH ikontáblázatát. Az adatlemezre kimentett fájlt a munka végeztével nevezzük át véletlen felülírás ellen.

Azt is megfigyeltem, hogy a program RESET esetén is képes tönkretenni magát a lemezen. Tehát ha RESET-eljük a programot, akkor is célszerű előzőleg kivenni a lemezt a meghajtóból!

Vicsotka Gyula

1. lista

```
10 PROGRAM "CHARTOLT.BAS"
20 REM KEYSOFT-1989.02.25.-
30 ALLOCATE 25
40 CODE LOADER=HEX$("F3,DB,B2,F5,
   3E,FF,D3,B2,FB,3E,6A,01,80,04,11,
   80,B4,F7,06,F3,F1,D3,B2,FB,C9")
50 OPEN #106:"LORI3" ACCESS INPUT
60 CALL USR(LOADER,0)
70 CLOSE #106
80 ! Innen folytathato a program!
```

2. lista

```
DI                  ;Megszakitas letiltas
IN A,(0B2h)         ;A Z-80-as 2-es lap
PUSH AF             ;tartalmanak elmentese
LD A,0FFh           ;A 2-es lapra a rendszer-
OUT (0B2h),A        ;szegmens belapozasa
EI                  ;Megszakitas engedelyezes
LD A,106            ;Csatorna szam
LD BC,1152          ;A fajl hossza
LD DE,0B480h        ;A karakter terulet RAM-ja
EXOS 6              ;Blokk olvasasa funkcio
DI
POP AF              ;A Z-80-as 2-es lap tar-
OUT (0B2h),A        ;talmanak eredeti allapota
EI
RET                 ;Vissza a BASIC-hez
```

Mi a manó?

Az „a" Stúdióval közösen meghirdetett ZZZIP pályázatunk továbbra is érvényben van. Számos érdekes játékprogramot kaptunk, tulajdonosaikat ZZZIP BASIC fordítóval jutalmaztuk. Azonban egyiket sem tudtuk átalakítás nélkül lefordítani. Ezért felhívjuk a pályázók figyelmét, hogy a ZZZIP egy egész értékű fordító. Így a BASIC programban lévő konstansok és változók csak egész értékűek lehetnek. Továbbá nem minden utasítást fogad el. Ezek a CHAIN, IMAGE, PRINT USING, TRACE, TYPE. A beépített függvényeket egész értékűnek tekinti. Amennyiben használni akarják, akkor kezdő értéküket arányosítani kell, azaz egész értékűvé kell transzformálni. Továbbra is várjuk pályázataikat, cikkeiket és észrevételeiket.

*

Varga Zoltán olvasónk BENCHMARK programokat futtatott. A BENCHMARK-ok általában olyan programok, amelyek a gép különböző tulajdonságait tesztelik az idő függvényében. A programokat és a többi gép adatait a szuper BIT-LET-ből ollózta.

	1	2	3	4	5	6	7	8
IS-BASIC	2,4	8,5	18,53	30,72	32,69	47,32	85,37	269,68
CP/M BASIC	2,31	5,71	15,47	16,22	17,9	29,68	46,22	81,72
C64	1,6	9,7	18,3	20,3	21,8	31,5	49,5	115,9
ZX-SPECTRUM	4,8	8,7	21,1	20,4	24	55,3	80,7	253
APPLE II	1,3	8,5	16	17,8	19,1	28,6	44,8	107

```
100 PROGRAM BEN_1
110 PRINT "S"
120 FOR K=1 TO 1000
130 NEXT K
140 PRINT "E"
150 END
```

```
100 PROGRAM BEN_2
110 PRINT "S"
120 LET K=0
130 LET K=K+1
140 IF K<1000 THEN 130
150 PRINT "E"
```

```
100 PROGRAM Ben_3
110 PRINT "S"
120 LET K=0
130 LET K=K+1
140 LET A=K/K*K+K-K
150 IF K<1000 THEN 130
160 PRINT "E"
170 END
```

```
100 PROGRAM BEN_4
110 PRINT "S"
120 LET K=0
130 LET K=K+1
140 LET A=K/2*3+4-5
150 IF K<1000 THEN 130
160 PRINT "E"
170 END
```

```
100 PROGRAM BEN_5
110 PRINT "S"
120 LET K=0
130 LET K=K+1
140 LET A=K/2*3+4-5
150 GOSUB 190
160 IF K<1000 THEN 130
170 PRINT "E"
180 STOP
190 RETURN
200 END
```

```
100 PROGRAM BEN_6
110 PRINT "S"
120 LET K=0
130 DIM M(5)
140 LET K=K+1
150 LET A=K/2*3+4-5
160 GOSUB 220
170 FOR L=1 TO 5
180 NEXT L
190 IF K<1000 THEN 140
200 PRINT "E"
210 STOP
220 RETURN
230 END
```

```
100 PROGRAM BEN_7
110 PRINT "S"
120 LET K=0
130 DIM M(5)
140 LET K=K+1
150 LET A=K/2*3+4-5
160 GOSUB 230
170 FOR L=1 TO 5
180   LET M(L)=A
190 NEXT L
200 IF K<1000 THEN 140
210 PRINT "E"
220 STOP
230 RETURN
240 END
```

```
100 PROGRAM BEN_8
110 PRINT "S"
120 LET K=0
130 LET K=K+1
140 LET A=K^2
150 LET B=LOG(K)
160 LET C=BIN(K)
170 IF K<1000 THEN 130
180 PRINT "E"
190 END
```

Gaetsch Günterné rajza

RAM-rezidens programok

Az MS-DOS Terminate-and-Stay-Resident rendszerhívása (Interrupt 21H Funkció 31H és az Interrupt 27H) lehetőséget biztosít a programozónak arra, hogy a RAM egy lefoglalt részén futtatható kódot, illetve adatokat helyezzen el, amelyek más programok futása alatt is a tárban maradnak. Ily módon RAM-rezidenssé tehetők globális adatok, megszakításkezelők, teljes alkalmazói programok. Az MS-DOS Terminate-and-Stay-Resident lehetőségét használó programokat nevezzük TSR-eknek.

A TSR felépítése

A TSR végrehajtható kódja, illetve adatai RAM-rezidens és RAM-tranziens részekre oszthatók fel:

Magasabb címek	Inicializáló kód és adat Alkalmazói program és adatai Monitorrutinok
Alacsonyabb címek	PSP

A TSR RAM-rezidens része a valamilyen hasznos funkciót megvalósító alkalmazói program végrehajtható kódját, illetve annak adatait tartalmazza. A tranziensz rész installálja a TSR-t: inicializálja a RAM-rezidens részen található változókat és megszakításkezelőket, majd végrehajtja az MS-DOS Terminate-and-Stay-Resident funkcióját, amely a memóriában hagyja a RAM-rezidens részt, és felszabadítja a tranziens rész által elfoglalt memóriát.

A TSR tranziens része akkor fut, amikor a programot tartalmazó .EXE vagy .COM file végrehajtódik; a RAM-rezidens rész viszont csak akkor, ha az éppen aktív program (foreground process) vagy valamilyen szoftver-, illetve hardvermegszakítás aktivizálja.

A TSR lehet aktív vagy passzív, attól függően, hogy a RAM-rezidens program milyen módon kapja meg a vezérlést.

Passzív és aktív TSR

A passzív TSR csak akkor aktivizálódik, ha valamilyen program átadja neki a vezérlést, akár szoftvermegszakítással, akár egy hosszú JMP-pal vagy CALL-lal. A TSR nem szakítja meg a hívó program futását, így mind az MS-DOS, mind a BIOS, mind pedig a hardver státusza jól definiálható abban a pillanatban, amikor a TSR megkapja a vezérlést.

Ezzel ellentétben az aktív TSR, akár az éppen futó programot is megszakítva, valamilyen külső esemény előfordulásakor aktivizálódik, például egy bizonyos billentyűkombináció vagy meghatározott hardvermegszakítás hatására. Ebből következően az aktív TSR majdnem mindig megszakítja valamilyen más program futását, és felfüggeszti annak működését. Elkerülendő a megszakított program összeomlását, az aktív TSR-nek felügyelnie kell az MS-DOS, a ROM BIOS és a hardver státuszát, és biztosítania kell, hogy a TSR csak akkor vegye át a vezérlést, amikor ezt biztonsággal megteheti.

A passzív TSR felépítése általában egyszerűbb, mivel a hívó program környezetében fut, feltételezi, hogy felhasználhatja a hívó program Program Segment Prefixét (PSP), nyitott fájljait, aktív könyvtárát és így tovább. Ilyenkor a hívó program felelős azért, hogy a hardver és az MS-DOS stabil állapotban legyen, mielőtt átadja a vezérlést a passzív TSR-nek.

Az aktív TSR viszont aszinkron módon aktivizálódik, ezért a hardver, az MS-DOS és az éppen futó program állapota meghatározhatatlan a TSR-t aktivizáló esemény bekövetkezésekor. Emiatt az aktív TSR felépítése sokkal bonyolultabb. Az aktív TSR RAM-rezidens részének tartalmaznia kell olyan modulokat, amelyek felügyelik az operációs rendszert, és meghatározzák, hogy a vezérlés mikor adható át a TSR alkalmazói program részének. Ezek a felügyelő rutinok tipikusan a billentyűzetbemenetet, a ROM BIOS funkcióinak kiszolgálását, a hardvermegszakítások kiszolgálását és az MS-DOS funkciók kiszolgálását ellenőrzik. A TSR csak akkor aktivizálja az alkalmazói programot (a RAM-rezidens résznek az a tagja, amely a TSR fő feladatát végzi), ha érzékeli a megfelelő billentyűkombinációt, és az alkalmazói program végrehajtása nem ütközik a megszakítások és az MS-DOS funkciók kiszolgálásával.

Az aktív TSR általában tartalmaz egy olyan RAM-rezidens modult, amely megvizsgálja a billentyűzetbemenetet: megfelel-e egy „hot-key"-nek nevezett billentyűkombinációnak. A felhasználó a RAM-rezidens programot ezzel a hot-key billentyűkombinációval indíthatja el. A TSR-ben alkalmazott technika — amely felügyeli a billentyűzetbemenetet — függ a billentyűzet hardvermegvalósításától. Az IBM PC és PS/2 családokban a billentyűzetben lévő processzor minden egyes billentyű leütésekor egy Ø9H sorszámú megszakítást generál. Ezáltal a TSR egy saját Ø9H-s megszakításkezelő segítségével felügyelheti a billentyűleütéseket, érzékelheti a meghatározott hot-key kombinációt.

A ROM BIOS-megszakítás kiszolgálása

Az IBM PC-k és PS/2-k ROM BIOS rutinjai nem reentránsak. A ROM BIOS-rutinokat hívó aktív TSR-nek biztosítania kell, hogy a TSR nem kísérel meg olyan funkciót végrehajtani, amelyet az éppen aktív program is abban a pillanatban hívott, amikor a TSR átvette a rendszer vezérlését. Mivel az IBM PC-n a ROM BIOS-rutinok szoftvermegszakítások segítségével aktivizálhatók, ezért az aktív TSR az eredeti kezelőket saját kezelőkkel helyettesítve felügyelheti a ROM BIOS állapotát.

Minden ilyen saját kezelő karbantart egy állapotjelzőt (flaget), amelyet a megfelelő ROM BIOS-hívás előtt megnövel, majd a ROM BIOS-rutin befejezésekor csökkent eggyel. Így a TSR felügyelő rutinjai az állapotjelző vizsgálatával eldönthetik, hogy nem reentráns ROM BIOS-rutin áll-e végrehajtás alatt. Ha az állapotflag alapértéke 0, akkor a flag 0-tól eltérő értéke jelzi, hogy a rendszer éppen ROM BIOS-szinten van.

Hardvermegszakítás és az MS-DOS-funkciók

Az aktív TSR felügyelő rutinjai — melyek maguk is hardvermegszakítás eredményeként aktivizálódnak — nem indíthatják el a TSR alkalmazói program részét, ha bármilyen hardvermegszakítás van kiszolgálás alatt.

Az IBM PC-ken a hardvermegszakítások kiszolgálása prioritásos sorrendben történik; ezt az Intel 8259A programozható megszakításvezérlő határozza meg. A 8259A nem engedélyez megszakításkiszolgálást, ha éppen egy előző, azonos vagy magasabb prioritásmegszakítás áll kiszolgálás alatt.

Minden hardvermegszakítás-kezelőnek tartalmaznia kell egy olyan részt, amely jelzi a 8259A áramkörnek a megszakításkiszolgálás befejezését. Ha a TSR megszakítja egy másik hardvermegszakítást kiszolgáló rutin végrehajtását — mielőtt az jelezhetné a 8259A-nak a megszakításkiszolgálás befejezését —, az a következő hardvermegszakítások letiltására vezethet. Az olyan nagy prioritású megszakítások letiltása, mint például a timeré (Interrupt Ø8H) vagy a billentyűzeté (Interrupt Ø9H), a rendszer összeomlását okozhatják. Ezért az aktív TSR-nek a 8259A lekérdezésével felügyelnie kell az összes hardvermegszakítás kiszolgálásának állapotát, biztosítva, hogy a RAM-rezidens alkalmazói programnak csak akkor adó-

dik át a vezérlés, amikor semmilyen hardvermegszakítás nem áll kiszolgálás alatt.

Ellentétben a ROM BIOS-rutinokkal, az MS-DOS korlátozott mértékben reentráns. Bizonyos esetekben az aktív program által hívott MS-DOS 21H megszakítás funkcióinak kiszolgálása felfüggeszthető úgy, hogy a RAM-rezidens alkalmazói program is hívhat 21H megszakításfunkciókat. Ezért az aktív TSR-nek felügyelnie kell az operációs rendszer tevékenységét, azért, hogy meghatározhassa, mikor van lehetősége a TSR alkalmazói programnak MS-DOS hívást végrehajtani.

TSR-funkciók

Az MS-DOS kétféle módon nyújt lehetőséget a kilépésre a futó programból, úgy, hogy annak egy részét rezidensen a RAM-ban hagyja. A javasolt mód a 21H megszakítás 31H funkciója.

Interrupt 21H Funkció 31H hívásakor a DX-ben lévő érték határozza meg annak a RAM-területnek a nagyságát (paragrafusokban) a Program Segment Prefixtől (PSP) kezdődően, amely a program kilépése után a tárban allokálva marad. A funkció abban hasonlít a 4CH funkcióhoz (Terminate Process With Return Code), hogy a visszatérési kódot itt is az AL regiszterbe kell tenni, azonban a nyitott fájlokat nem zárja le automatikusan.

A *27H megszakítás* esetén a DX regiszterbe elhelyezett érték határozza meg a RAM-rezidens program memóriaigényét bájtokban kifejezve. Az MS-DOS a DX-ben lévő értéket paragrafussá konvertálja, nullázza az AL regisztert és a vezérlést a 21H megszakítás 31H funkcióját kiszolgáló részre adja át. A 27H megszakítása kevésbé rugalmas, mint a 21H megszakítás 31H funkciója, mivel a RAM-rezidens program mérete korlátozott (64 k), a CS regiszternek a PSP szegmenscímét kell tartalmaznia, és nincs lehetőség visszatérési kód átadására. Az MS-DOS későbbi verziói elsődlegesen az 1.x verziókkal való kompatibilitás miatt támogatják a 27H megszakítás alkalmazását.

RAM-szervezés

A TSR által explicit módon allokált RAM-területen kívül — melynek nagyságát a DX regiszter tartalma határozza meg — az MS-DOS külön RAM-ot foglal le a TSR környezete számára, amely szintén rezidens marad a TSR installációs részének befejeződése után. (A környezet paragrafuscímét a TSR PSP-jének 2CH ofszetjén lehet megtalálni.) Ezenkívül, ha a TSR installációs része a 21H megszakítás 48H funkciójával (Allocate Memory Block) további RAM-területet foglal le, ez a terület is foglalt marad a program kilépése után. Ha a RAM-rezidens programnak nincs szüksége erre a járulékos RAM-területre, akkor a TSR installációs része fel is szabadíthatja azt a 21H megszakítás 49H funkciójával (Free Memory Block), mielőtt végrehajtaná a rezidens kilépést (21H megszakítás 31H funkció).

Két — 21H megszakításhoz kapcsolódó — funkcióhívás áll rendelkezésre a 8086 család megszakításvektorainak lekérdezésére, illetve módosítására. A 25H funkció (Set Interrupt Vector) módosítja az AL regiszterben megadott sorszámú megszakítás vektorát a DS : DX-ben megadott szegmens- és ofszetértékre.

A 35H funkció (Get Interrupt Vector) ennek fordítottját végzi: az AL regiszter által meghatározott sorszámú megszakítás vektorát tölti be az ES : BX regiszterpárba. Bár lehetőség van a vektorok direkt manipulálására, a 21H megszakítás 25H és 35H funkcióinak használata általában sokkal kényelmesebb, és biztosítja a kompatibilitást az MS-DOS későbbi verzióival.

1. TÁBLÁZAT

```
MS-DOS 2.x verzió

                if   (FunkcióSzám >= 01H and FunkcióSzám <= 0CH)
                     or
                     FunkcióSzám = 50H
                     or
                     FunkcióSzám = 51H

                then if ErrorMode = 0
                     then IOStack
                     else AuxStack

                else ErrorMode = 0
                     DiskStack

MS-DOS 3.0 verzió

                if   FunkcióSzám = 50H
                     or
                     FunkcióSzám = 51H
                     or
                     FunkcióSzám = 62H

                then a hívó program stack-je

                else if ( FunkcióSzám >= 01H and FunkcióSzám <= 0CH)
                        or
                        FunkcióSzám = 59H

                     then if ErrorMode = 0
                          then IOStack
                          else AuxStack

                     else ErrorMode = 0
                          DiskStack

MS-DOS 3.1 és későbbi verziók

                if FunkcióSzám = 33H
                   or
                   FunkcióSzám = 50H
                   or
                   FunkcióSzám = 51H
                   or
                   FunkcióSzám = 62H

                then a hívó program stack-je

                else if ( FunkcióSzám >= 01H and FunkcióSzám <= 0CH)
                        or
                        FunkcióSzám = 59H

                     then if ErrorMode = 0
                          then IOStack
                          else AuxStack

                     else if FunkcióSzám = 59H
                          then AuxStack
                          else ErrorMode = 0
                               DiskStack
```

Az MS-DOS a program PSP-jét olyan, a program szempontjából egyedi információk nyomon követésére használja, mint például a parancssor-paraméterek, a program környezetének szegmenscíme. Ezen információk elérése céljából az MS-DOS tartalmaz egy belső változót, amely mindig az éppen aktív program PSP-jének címét tartalmazza. Amikor egy RAM-rezidens alkalmazói program aktivizálódik, az 51H (Get Program Segment Prefix) és az 50H (Set Program Segment Prefix) funkciók segítségével mentheti el a belső változó aktuális tartalmát, illetve módosíthatja úgy, hogy az saját PSP-jének címét tartalmazza. Az 50H funkcióval módosítható az MS-DOS belső változójának tartalma, amely az éppen aktív program PSP-jének címét tartalmazza. Az 51H funkció a belső MS-DOS-változó tartalmával tér vissza.

2. TÁBLÁZAT

NÉV	HÍVÁS	VISSZATÉRÉS
Terminate and Stay Resident	AH = 31H AL = visszatérési kód DX = a rezidens program mérete (paragrafusokban) INT 21H	----------
Terminate and Stay Resident	CS = PSP DX = a rezidens program mérete (bájtokban) INT 27H	----------
Set Interrupt Vector	AH = 25H AL = megszakítás sorszám DS:DX = megszakítás kezelő kezdőcíme INT 21H	----------
Get Interrupt Vector	AH = 35H AL= megszakítás sorszám INT 21H	ES:BX=megszakítás kezelő kezdőcíme
Set PSP Address	AH = 50H BX = PSP szegmens cím INT 21H	----------
Get PSP Address	AH = 51H BX = PSP szegmens cím INT 21H	
Set Extended Error Information	AX = 5D0AH DS:DX = egy 11 szavas struktúra címe 0.szó:AX regiszter az 59H funkcióból való visszatérés után 1.szó:BX regiszter 2.szó:CX regiszter 3.szó:DX regiszter 4.szó:SI regiszter 5.szó:DI regiszter 6.szó:DS regiszter 7.szó:ES regiszter 8-0AH szó:fenntartott, értéke kötelezően 0 INT 21H	----------
Get Extended Error Information	AH = 59H BX = 0 INT 21H	AX = kiterjesztett hibakód BH = hibaosztály BL = javasolt tevékenység CH = hiba helye
Set Disk Transfer Area Address	AH = 1AH DS:DX = DTA cím INT 21H	----------
Get Disk Transfer Area Address	AH = 2FH INT 21H	ES:BX = aktív DTA címe
Get InDOS Flag Address	AH = 34H INT 21H	ES:BX = az InDOS flag címe

A kiterjesztett hibakód

Az MS-DOS 3.1 és későbbi verzióiban a RAM-rezidens program megőrizheti az aktív program kiterjesztett hibainformációit, mivel ha a RAM-rezidens alkalmazói program MS-DOS-hibával találkozik, az aktív programhoz tartozó kiterjesztett hibainformáció még elérhető és helyreállítható. Az 59H és az 5D0AH funkciók biztosítják ezeknek az információknak az elmentését és helyreállítását a TSR alkalmazói program futása alatt. Az 59H funkció (Get Extended Error Information), amely a 3.0 verziótól kezdődően érhető el, a legutóbbi MS-DOS hibát leíró részletes információkkal tér vissza. Ezzel ellentétes műveletet végez az 5D0AH-s funkció (Set Extended Error Information), amely csak a 3.1 és későbbi verziókban használható. Ez a funkció a hívó programban definiált adatstruktúrát másolja be az MS-DOS egyik belső táblázatába.

Számos MS-DOS adatátviteli funkció, nevezetesen a 21H, 22H, 27H és 28H (Random Read és Random Write), valamint a 14H, 15H funkciók (Sequential Read és Sequential Write) a program által meghatározott lemezátviteli területet (DTA) igényelnek. Alapértelmezés szerint a program DTA-ja a PSP 80H ofszetjétől kezdődően helyezkedik el. Ha a RAM-rezidens alkalmazói program DTA-t használó MS-DOS funkciót hív, akkor a TSR-nek a megszakított program DTA-jának címét a 21H megszakítás 2FH funkciójának segítségével (Get Disk Transfer Area Address) el kell mentenie, az 1AH funkcióval (Set Disk Transfer Area Address) be kell töltenie saját DTA-jának címét, majd pedig a kilépés előtt helyre kell állítania az eredeti DTA címet.

Az MS-DOS üres megszakítása (Interrupt 28H)

Az első 12 MS-DOS funkciónak (01H...0CH) várakoznia kell valamilyen esemény bekövetkezésére. Például az XXH funkció addig várakozik, amíg a felhasználó le nem üt egy billentyűt. Ezek a funkciók tartalmaznak egy „vak ciklust", amely addig folytatódik, amíg a meghatározott esemény be nem következik. Azért, hogy a várakozás alatt más tevékenységet is el lehessen végezni, ezek az MS-DOS funkciók a várakozó cikluson belül tartalmaznak egy 28H-s megszakításhívást. Az alapértelmezés szerint a 28H megszakításkezelő egy IRET utasítást tartalmaz. Amennyiben a 28H megszakításvektorába saját kezelőjének címét tölti be, a TSR elvégezhet néhány hasznos műveletet az idő alatt, amelyet az MS-DOS egyébként üres várakozással töltene. Például egy saját 28H megszakításkezelő megvizsgálhatja a rendszer aktuális állapotát, és így eldöntheti,

hogy a RAM-rezidens alkalmazói program biztonsággal aktivizálható-e vagy sem.

A TSR az MS-DOS aktuális állapotára annak belsőverem- (stack-) használatából, illetve egy belső flag-párból következtethet. Ezek a státuszinformációk alapvető fontosságúak az aktív TSR számára, mivel a RAM-rezidens alkalmazói program csak akkor hajthat végre MS-DOS-hívást, ha ezek a hívások nem szakítják meg az éppen aktív program egy korábbi, hasonló hívását.

Az MS-DOS 2.0 és későbbi verziói három belső vermet használhatnak: Az IOStacket, a DiskStacket és a kiegészítő AuxStacket. Általában az IOStacket a 21H megszakítás Ø1H...ØCH-s funkcióit, a DiskStacket pedig a többi 21H-s megszakításfunkció használja. Az MS-DOS az AuxStacket csak akkor használja, ha kritikus hibát érzékel, és utána kiadja a 24H-s megszakításhívást. Az MS-DOS belsőveremhasználata az MS-DOS funkció sorszámától és a kritikus hiba flag-értékétől függ.

A kritikus hiba jelzése

A kritikus hibát jelző (ErrorMode) 1 bájt hosszú flag értéke — amelyet az MS-DOS a kritikus hiba előfordulásának jelzésére használ — normális, hibamentes végrehajtás esetén Ø.

Amikor az MS-DOS kritikus hibát érzékel a 24H megszakítás meghívása előtt, a flaget valamilyen nullától eltérő értékre állítja be. Ha a 24H megszakításkezelő a 21H megszakítás használatával valamilyen MS-DOS funkciót hív, az ErrorMode nullától eltérő értéke fogja jelezni az MS-DOS-nak, hogy a 21H megszakítás Ø1H...ØCH funkcióinak hívásakor a normális B/K verem helyett a kiegészítő vermet használja. Más szóval, amikor a vezérlés a 21H megszakításon keresztül az MS-DOS-nak adódik át, az adott funkción belül a funkció sorszáma és a kritikus hiba flag-értéke együttesen határozza meg az MS-DOS verem használatát.

Az *1. táblázat* mutatja az adott funkcióhoz tartozó verem kiválasztásának algoritmusát, a funkcióba való belépéskor. Tehát a Ø1H...ØCH funkciók esetén az MS-DOS az IOStacket használja, ha a kritikus hiba flag-értéke Ø, a flag nullától eltérő értéke esetén (kritikus hiba) pedig az alkalmazott verem az AuxStack. A ØCH-nál nagyobb sorszámú funkcióknál az MS-DOS általában a DiskStacket használja, azonban az 50H, 51H és 59H sorszámú funkciók fontos kivételek. Az 50H és 51H funkciók vagy az IOStacket (2.x verziók) vagy a hívó program saját vermét (3.x verziók) használják. A 3.0 verzióban az 59H funkció vagy az IOStacket vagy az AuxStacket használja a kritikus hiba flag-értékétől függően, de a 3.1-es és későbbi verzióknál az 59H funkció mindig az AuxStacket használja.

A táblázat alapján látható, hogy a Ø1H...ØCH funkciók korlátozott mértékben reentránsak, így tehát a kritikus hibát jelző flag nullától eltérő értéke esetén, a kritikus hibát kezelő rutin (Interrupt 24H) még használhatja ezeket a funkciókat. Ugyanis ilyenkor, mivel a flag nem nulla, a Ø1H...ØCH funkciók az IOStack helyett az AuxStacket használják.

Ezért, ha a kritikus hiba észlelésekor az MS-DOS az IOStacket használja (Ø1H...ØCH funkciók), ennek tartalma nem változik meg, ha a hibakezelő meghívja a Ø1H...ØCH funkciókat. A veremválasztás logikája némileg különböző az MS-DOS 2-es és 3-as verziója között. A 3.x verziókban néhány funkció, nevezetesen az 50H és 51H, egyáltalán nem használja az MS-DOS vermeit. Ezek a funkciók kevésbé bonyolult feladatokat hajtanak végre, melyeknek veremigénye minimális, és így elegendő számukra a hívó program saját verme is.

Az InDOS flag

Az InDOS egy 1 bájt hosszú flag, amely minden 21H-s funkcióhívás esetén eggyel nő, és a funkció befejeződésekor eggyel csökken. A flag értéke mindaddig nullától eltérő marad, míg a rendszer MS-DOS szinten van, azaz az MS-DOS-on belüli kód áll végrehajtás alatt. Az InDOS értéke nem jelzi azt, hogy az MS-DOS melyik belső vermet használja. Amikor az MS-DOS kritikus hibát érzékel, a 24H megszakítás meghívása előtt nullázza az InDOS-t. Erre az olyan kritikus hibakezelőkkel való együttműködés miatt van szükség, amelyek a vezérlést nem adják vissza az MS-DOS-nak. Ha ugyanis az InDOS flag értéke nem lenne nulla, amikor egy ilyen kritikus hibakezelő veszi át a vezérlést, a flag értéke soha nem csökkenne, értéke egy újabb MS-DOS-hívás esetén hibás lenne.

Az 1 bájtos InDOS flag címét a 21H-s megszakítás 34H funkciójával (Return Address of InDOS Flag) kaphatjuk meg.

A 3.1 és ennél későbbi verziókban a szintén 1 bájtos ErrorMode az InDOS-t megelőző bájton helyezkedik el, és így az említett funkció használatával mindkét flag címe megkapható. Sajnos az ezt megelőző verziókban nincs ilyen egyszerű lehetőség a kritikus hibát jelző flag címének meghatározására. A javasolt technika az MS-DOS szegmens letapogatása (a szegmens kezdőcímét a 34H funkció szolgáltatja az ES regiszterben), keresve az alábbi utasításokat:

```
test    ss: [CriticalErrorFlag], 0FFH;
        (3.1 és később)
jne     NearLabel
push    ss: [NearWord]
int     28H
vagy
cmp     ss: [CritialErrorFlag], 00H;
        (3.1-nél korábbi)
jne     NearLabel
int     28H
```

Ha megtaláljuk a TEST vagy CMP utasítást, a kritikus hiba címe meghatározható az utasítás operandusmezőjéből.

A multiplex megszakítás

Az MS-DOS multiplex megszakítás (Interrupt 2FH) segítségével ellenőrizheti a program egy TSR jelenlétét, illetve ez szolgál a TSR-rel való kommunikációra is. A program úgy kommunikál a TSR-rel, hogy az AH regiszterbe egy azonosító értéket (ID), az AL regiszterbe pedig egy funkciókódot helyez el, majd kiadja a 2FH megszakítást. A TSR 2FH megszakítás kezelője összehasonlítja az AH-ban lévő értéket a saját, előre meghatározott ID értékével. Ha megegyeznek, a TSR magánál tartja a vezérlést, és végrehajtja az AL által specifikált funkciót. Ha az értékek nem egyeznek meg, a TSR átadja a vezérlést az előzőleg installált 2FH-s megszakításkezelőnek. (A multiplex ID értékek a ØØH...7FH tartományban az MS-DOS használatára vannak fenntartva; a felhasználói multiplex ID-knek a 80H...ØFFH tartományba kell esniük.)

A következő példában a multiplex megszakításkezelője csak az AL=ØØH-hoz tartozó funkciót ismeri fel. Ebben az esetben a kezelő az AL-ben ØFFH értékkel tér vissza, jelezve, hogy a kezelő valóban RAM-rezidens. Így a program ellenőrizheti a kezelő jelenlétét, úgy, hogy a megfelelő ID, illetve AL=ØØH értékkel kiadja a 2FH megszakításhívást.

```
mov     ah,MultiplexID
mov     al,ØØH
int     2FH
comp    al,ØFFH
je      AlreadyInstalled
```

Az azonosító értéke egyediségét biztosítani kell, az ID-értéket a TSR installációjával egy időben kell meghatározni. Erre egy lehetőség az ID-érték átadása parancssor-paraméterként, a TSR installálásakor. Az azonosító érték átadható egy környezeti változóban is. Így az ID-érték megtalálható mind a TSR, mind a TSR jelenlétét a 2FH megszakítás segítségével ellenőrző más programok környezetében is.

Az MS-DOS Terminate-and Stay-Resident utilitykat támogató szolgáltatásait a *2. táblázat* tartalmazza.

Rendezvény

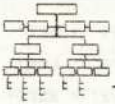
A Híradástechnikai Tudományos Egyesület Mikroszámítógépes Programnyelvek és Operációs Rendszerek Szakosztálya a SOTE I. sz. Anatómiai Intézetével közösen vitát rendez a mesterséges intelligencia jelentéséről és felhasználásáról. A vita a tavaszi szakosztálygyűlés folytatása. Felkért hozzászóló dr. Lábas Elemér professzor, A természetes és mesterséges értelem c. könyv szerzője.

A rendezvény időpontja: *1989. nov. 20., 14 óra.*

Helye: *MTESZ Székház, Bp. V., Kossuth tér 6—8. III. em. 333-as terem.*

Minden érdeklődőt szívesen látnak a szervezők.

VEGYES



Nyelvek és betűk

Tud-e a DOS magyarul?

A számítógépek MS-DOS operációs rendszere amerikai találmány. Írói nem gondoltak arra, hogy akik e berendezéseket használni fogják, azok nem mind értenek angolul (amerikaiul). Szerencsére viszonylag hamar nyilvánvalóvá vált, hogy szükségük van más nyelvek ékezetes betűire, más jelekre. Elegendő ránézni kinek-kinek arra a billentyűzetre, amelyen dolgozik! Megtalálja-e rajta ezeket a jeleket:

½ ¼ « » í ó ú ñ Ñ ª º ¿ ù ÿ ö Ü £ ₧ ƒ

Ugye nem mindegyiket látja ott, pedig szüksége lenne rájuk.

Nos, ezeket a karaktereket minden gép ismeri, legalábbis akkor, ha nem tettek bele más, az IBM szabványtól eltérő, „magyarított" karakter-ROM-ot. A nemzeti karakterkészletekre vonatkozó igényt az újabb DOS-verziókban figyelembe vette az IBM, s elkészítették a billentyűzet-átdefiniáló külső DOS-utasításokat, egyszerűbben mondva a tárrezidens kiegészítő programokat. Újabb karakterekkel ezek sem tudják kibővíteni a gépet, de a karaktergenerátor-ROM-ból be tudják tölteni a billentyűzet megfelelő helyére azokat a karaktereket, amelyek a ROM-ban benne vannak, s az IBM úgynevezett nemzetközi karakterkészletébe tartoznak.

Ugyanakkor történeti okai vannak, miért nem bővíthető minden korláton túl a szokásos billentyűzetkiosztás. Több magyar vállalkozás próbálta meg úgy átírni a ROM megfelelő rutinjait, hogy nagyobb gombszámú billentyűzetet is tudjon fogadni. A kísérlet a gyakorlatban siralmas „eredményt" produkált: az IBM gépből éppen a legfontosabb tulajdonságot, az univerzalitást áldozta fel. Azt, hogy egy IBM-kompatibilis gépen megírt program — megfelelő grafikus kártya és operációs rendszer esetén — változtatás nélkül fut a másikon.

A DOS-ban lévő KEYB**.COM utasításoknál fontos tudnivaló a következő néhány alapelv, amelyet minden változtatásnál figyelembe kell venni (egészen addig, amíg nem a magyar csúcstechnológia diktálja a szabványt a világpiacon).

Amerikából jöttem...

1. A DOS alapértelmezése az amerikai nemzeti ábécé, karakterkészlete pedig az IBM International Character Set szabványnak felel meg, amely tartalmazza sok nemzeti ábécé karakterkészletét is. Ezekből hozzánk a svéd ábécé áll a legközelebb. Legkényelmesebben viszont a német billentyűzet használható, mert ismeri a repülő ékezetet, s ezzel a legtöbb hazai betű előállítható.

2. A DOS alapértelmezése szerint a billentyű-átdefiniáló tárrezidens program kötelező elnevezési szabálya:

KEY***.COM,

ahol ***helyett annak az országnak a rövidítése van, amelynek a karakterkészletére definiáljuk át a billentyűzetet. Ez utóbbi helyett szabadon választott rövidítés is állhat akkor, ha egy programhoz illesztettük a klaviatúránkat. Példák:

KEYBUS.COM	= USA/Ausztrália
KEYBUK.COM	= Anglia
KEYBFR.COM	= Franciaország
KEYBIT.COM	= Olaszország
KEYBGR.COM	= NSZK

KEYBHUN.COM szerinti klaviatura

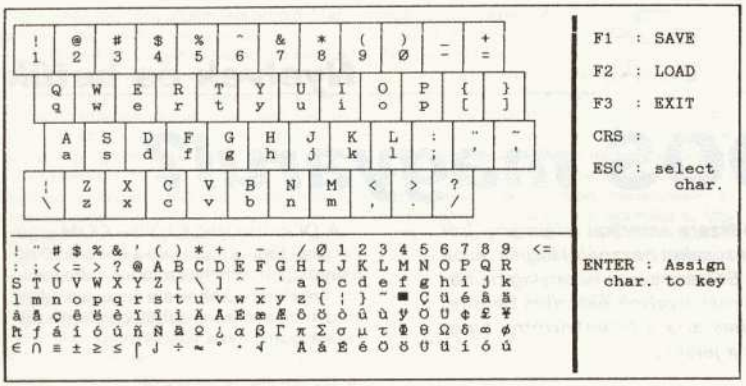
! 1	@ 2	# 3	$ 4	% 5	^ 6	& 7	" 8	(9	) Ø	_ ó	ú =
Q q	W w	E e	R r	T t	Y y	U u	I i	O o	P p	Ä á	É é
A a	S s	D d	F f	G g	H h	J j	K k	L l	: ;	Ö ö	Ü ü
í \	Z z	X x	C c	V v	B b	N n	M m	< ,	> .	? /	

1. ábra

2. ábra

KEYBHU.COM szerinti klaviatura

Ä ä	" 2	§ 3	= 4	% 5	! 6	& 7	(8	) 9	Ö ö	Ü ü	ò ó
Q q	W w	E e	R r	T t	Z z	U u	I i	O o	P p	ÿ ô	Æ ú
A a	S s	D d	F f	G g	H h	J j	K k	L l	É é	Á á	}]
{ [	Y y	X x	C c	V v	B b	N n	M m	; ,	: .	? /	



3. ábra

KEYBSP.COM = Spanyolország
KEYBDA.COM = Dánia
KEYBNO.COM = Norvégia
KEYBCHG.COM = Svájc (német klaviatúra)
KEYBCHF.COM = Svájc (francia klaviatúra)

és végül a mi karakterkészletünk:
KEYBHU.COM = Magyarország
KEYBHUN.COM = Magyarország

A magyar ékezetes karakterkészlet, a keybhu.com szokásos klaviatúrakiosztása csak részben követi az írógépszabványt *(1. ábra)*, csakúgy, mint a keybhun.com-é *(2. ábra)*.

3. A DOS ekkor azt használja ki, hogy nyolc biten — mert ennyit használ egy betű gépi megjelenítésére — 256 karakter helyezhető el. Ebbe kell beleférniük a félgrafikus jeleknek, a CTRL, valamint az ALT billentyűk kombinációjának. Így alakult ki az a gyakorlat, hogy az amerikai billentyűzet karaktereit nemzetközi karakterkészletnek — bár láttuk, ezek valójában az USA nemzeti karakterei —, a többi átdefiniált karakterkészletet pedig nemzeti karakterkészletnek nevezik. A karaktergenerátor teljes szabványos jelkészletét „Extended International Character Set IBM Table Latin II"-nek nevezzük. (E csúnya kifejezés szabatos magyar fordítása: Kiterjesztett nemzetközi karakterkészlet az IBM latin betűs II. karaktertáblázata szerint.)

4. A DOS szerinti szabványnak megfelelő billentyű-átdefiniáló program szabványosan-kötelezően mindig két karakterkészletet kezel:

a) a DOS alapértelmezése szerinti amerikai billentyűzetet,

b) az átdefiniált nemzeti billentyűzetet. A billentyűzetkezelő programot vagy az Autoexec.bat fájlból indítjuk, vagy pedig külön:

c: Keybhu ≪enter≫

Ekkor gépünk az átdefiniált karakterkészlettel fut, amelyből minden esetben átkapcsolhatunk az alapértelmezés szerinti amerikai betűkészletbe az

≪ALT≫ ≪F1≫

billentyűk együttes benyomásával, vissza pedig az

≪ALT≫ ≪F2≫

billentyűk együttes benyomásával.

Inkább ne bütyköljünk!

5. A DOS szerinti szabványnak megfelelő átdefiniáló program a bütykölt ROM-karaktergenerátor esetén kétféle hibaüzenetet adhat angolul:

Internal table do not match

vagy

Unable to load key map routine.

Az első esetben a karaktergenerátort nem a DOS szabályainak megfelelően módosították; a másik esetben a DOS a tasztatúrát másképpen „látja", mint ahogy a billentyűzetkezelő rutin várja. Ezekben az esetekben csak a megfelelő KEYB**.COM segíthet — ha egyáltalán segíthet —, ha nem nagy az inkompatibilitás, s a magyar karakter nem vezérlőkód helyére került a nagy igyekezetben. A nyomtatók nemzetközi karakterkészletet állító kapcsolói a nyomtató ROM-ját összehangolják a megfelelő KEYB**. COM definícióval, de eredeti magyar KEYBHU.COM és természetesen ennek megfelelő nyomtató-ROM az eredeti DOS-ban nem létezik.

6. Magyarországon igen sokan használják — bár tudomásunk szerint nem szabad szoftver — a KLAVGEN.EXE nevű karakterkészlet-definiáló szoftvert, amelyet a felcím szerint két magyar programozó, Dozmáti és Szinetár készített 1985-ben. Egyike a legjobban alkalmazható szoftvereknek, segítségével mindenki nagyon egyszerűen elkészítheti a kedve szerinti KEYB**.COM-ot. Mivel a szoftver által adott segítség nem egyértelmű, az alábbiakban szeretnénk röviden ismertetni használatát. Fontos tudnivaló, hogy csak a szabványos IBM billentyűkre lehet vele átdefiniáló programot készíteni, nem alkalmas a szabványosnál több gombbal készített, „bütykölt", egyedi szabványú klaviatúrák újradefiniálására.

A program indítása után egy rövid leírással jelentkezik be. Ebben arra hívja fel a figyelmünket, hogy eredményül két fájlt kapunk, amelyek azonos nevűek, de különböző kiterjesztésűek. Ebből a *******.COM a futtatható, a *******.KLV arra szolgál, hogy ha a későbbiekben újra át akarjuk alakítani a már elkészült programunkat, ez lesz a forrásállomány. Amennyiben kiindulásul nincs

CWI-táblázat

Alt+decimális ért	HEXA	Eredeti karakterkép	Magyar új karakter
129	81	ü	rövid ü /kicsi/
130	82	é	é /kicsi/
141	8D	ì	hosszú I /nagy/
143	8F	Ä	Ä /nagy/
144	90	É	É /nagy/
147	93	ô	hosszú ö /kicsi/
148	94	ö	rövid ö /kicsi/
149	95	ò	hosszú ó /nagy/
150	96	û	hosszú ü /kicsi/
151	97	ù	hosszú ú /nagy/
152	98	ÿ	hosszú ü /nagy/
153	99	Ö	rövid ö /nagy/
154	9A	Ü	rövid ü /nagy/
160	A0	á	á /kicsi/
161	A1	í	hosszú í /kicsi/
162	A2	ó	hosszú ó /kicsi/
163	A3	ú	hosszú ú /kicsi/
167	A7	º	hosszú ö /nagy/

ilyen állományunk, akkor az Input filename: .KLV kérdésre adott ENTER gombnyomással a DOS alapértelmezésének megfelelő amerikai karakterkészlet töltődik be *(3. ábra)*. A sematikus billentyűzetábra alatt pedig gépünk karakter-ROMjának teljes készletét láthatjuk, amelyből szabadon választhatunk. Az ábrán látható kép akkor jelenne meg, ha forrásfájl betöltése nélkül kezdtük volna el az új billentyűzet megszerkesztését.

Ha van már .KLV állományunk, az Input filename felszólításra be kell gépelni, természetesen kiterjesztés nélkül. Ha mentés nélkül szeretnénk abbahagyni a munkát, akkor az F3 gombbal kiléphetünk, miután y-nal válaszoltunk a program kérdésére.

Mit mond az „álmoskönyv"?

A program használata egyszerű; a nyilakkal és az < ENTER > gombbal választhatjuk ki a megfelelő karaktert, amit aztán a megfelelő helyre pozicionálunk. Az aktív mezőket (hogy a billentyűzeten vagy a karaktertáblában vagyunk-e) a < = jelzi. A mentés az F1 gombbal történik, ekkor kell megadni, milyen néven kívánjuk tárolni programunkat.

A megfelelő *******.COM fájl nevének begépelése után már használhatjuk az újonnan átdefiniált billentyűzetet. Ne feledjük, hogy csak az eredetileg beégetett billentyűzetre tudunk visszatérni a < CTRL > < ALT > < F1 > gombok egyidejű lenyomásával, s vissza pedig a < CTRL > < ALT > < F2 >-vel. Az „álmoskönyv nem javasolja" több billentyűzet-átdefiniáló program egymásra hívását, tekintettel arra, hogy tárrezidens programokról van szó.

A DOS szerinti szabványnak megfelelő karaktertáblázat sajnos nem tartalmaz minden magyar ékezetes karaktert. Így óhatatlanul bele kell nyúlni a karaktergenerátorba. Ennek viszont úgy kell megtörténnie a DOS szabályainak figyelembevételével, hogy a programok változatlanul megtalálhassák a nemzetközi kiterjesztett karakterkészlet jeleit. Erre hazánkban egyedül a Computerworld-Számítástechnika c. lap adott elfogadható konstruktív alternatívát: olyan szabványt dolgozott ki, amely a kulcsfontosságú jeleket nem bántja. Mindenképpen elterjesztésre érdemes a sokféle és nagyon sok program-összeférhetetlenséget okozó bütykölt, egyedi karaktergenerátorokkal szemben. Tervezetüket a *táblázatban* foglaljuk össze, amely a rövidség okán csak az IBM Extended Character Set Latin II-től való eltéréseket tartalmazza.

A táblázat a lehetőségekhez képest illeszkedik a DOS logikájához, nem hágja át annak sem a korlátait, sem a szabályait. Alkalmazásánál az egyes eredeti programok futtatása esetén kisebb a valószínűsége, hogy problémát okoz a felhasználónak, nem úgy, mint egyes, napjainkban kapható, magyarul beszélő, de a karakter-ROM cseréjét igénylő programok, amelyek lehetetlenné teszik más szoftverek alkalmazását.

(A Floppy lap nyomán)

AJÁNDÉK

Adatátvitel és terminálfunkció

Soros vonal a PC-n

Ismeretes, hogy az IBM PC soros vonala révén létesíthető kapcsolat más számítógépekkel, de nem könnyű a dolga annak, aki ezt a soros portot saját programjával szeretné használni. Az átvitel megvalósításakor a soros vonal adatvezetékei mellett a modemet vezérlő vonalak állapotát is figyelembe kell venni. Az IBM PC ROM BIOS-a alapszinten támogatja a soros port kezelését, de mértékadó vélemények szerint ez a ROM BIOS egyik leggyengébben megírt része.

A magas szintű nyelvek közül közvetlenül csak a BASIC-ben lehetséges a soros port használata, ott is csak igen korlátozott adatátviteli sebességek esetén.

A megoldandó feladat: egy Z80-alapú mikrogép soros vonalon való összekötése egy IBM XT-vel.

Maga a rendszer egy *IBM PC számítógépből* és a hozzá soros vonalon kapcsolódó *fejlesztendő rendszerből* áll. A Z80-alapú fejlesztendő rendszer indulásakor egy olyan EPROM-ot tartalmaz, amelyben egy, a programfuttatást segítő monitorprogram van; ez a program biztosítja, hogy az IBM PC-n fejlesztett programunkat hexadecimális formában töltsük a fejlesztendő rendszerünk RAM-jába.

Kettős szerepben

Fejlesztői üzemmódban a program szövegszerkesztővel írható, szerkeszthető, cross-assemblerrel fordítható, linkelhető, majd a tárgykód hexadecimális formában a rendszerbe tölthető, és a program a monitor üzemmódban futtatható.

Monitor üzemmódban a PC a Z80-as rendszer ki-bemeneti egységeként (termináljaként) működik, a monitornak a billentyűzeten keresztül parancsok adhatók, illetve a monitor válaszai a képernyőn jelennek meg. Természetesen a rendszerhez kell tartoznia egy EPROM-égető készüléknek is, amivel a végső fázisban a kifejlesztett és RAM-ban kipróbált programunkat beégethetjük. A PC-n a TERMINAL.EXE programot kell meghívni, aminek segítségével kommunikálni lehet a fejlesztendő rendszerrel. A TERMINAL program kezeli a soros adatátviteli vonalat és biztosítja a két üzemmód kezelését.

A két egység között három vezetéken folyt az adatátvitel, a modemvezérlő jelek nem voltak bekötve. A terminálfunkciót eredetileg egy QuickBasicben készült program valósította meg. Az adatáramlás nem volt túl gyors (1200 baud), de a BASIC program még így is elvesztett adatokat, ha azok túl nagy számban érkeztek.

A fenti problémák kiküszöbölésére kellett megvalósítani a 9600 baudos átvitelt egy új terminálprogram segítségével. Ez akkor igen előnyös, ha hosszú programokat kell a fejlesztendő rendszerbe tölteni.

A Z80-as rendszerben lévő soros port képes az ilyen adatátviteli sebességre, ott csupán az EPROM-ban kell átírni a baud-rate értékét.

A gond a PC oldalán jelentkezik. A bevezetőben is említett nehézségek természetesen nem hardverjellegűek. A ROM BIOS nem támogatja kellőképpen a soros vonalat. Alaphelyzetben a soros vonalnak csak egy egybájtos puffer áll rendelkezésre a vett adatok ideiglenes tárolására. Ez nagyon könnyen, még a kiürítése előtt felülíródik, adatvesztést okozva. A modemvezérlő jelek figyelését sem lehet letiltani a BIOS 14H megszaki-

tással (ezen keresztül érhetők el a soros vonalat támogató rutinok). A soros vonal megszakításos kezelését sem biztosítja a 14H megszakítás.

Segít a Dr. Dobb's

A megoldást a szerencse adta a kezünkbe. A Dr. Dobb's Journal 1987. júniusi számának 42—77. oldalán leírtak egy EXCOM (EXtended COM port driver=kiterjesztett kommunikációsport-vezérlő) nevű programot, ami a ROM BIOS összes hiányosságát kiküszöböli olyan módon, hogy kompatibilis marad az eredeti funkciókkal, de ezenkívül egyéb szolgáltatásokat is tartalmaz.

Ha az EXCOM-ot egyszer lefuttatjuk, memóriarezidenssé válik, és átveszi a soros vonal megszakításos és pufferelt kezelését. Ettől kezdve az összes programot, amely az INT 14H-t használja, az EXCOM fogja kiszolgálni. Ilyenkor átirányítja a 0BH, 0CH és 14H megszakításokat is. A 0BH és 0CH hardvermegszakítások akkor aktivizálódnak, amikor karakter érkezik vagy lehet küldeni a következőt. Ilyenkor történik a szükséges műveletek elvégzése is. A 14H szoftvermegszakításon keresztül hívhatjuk a megnövelt bemeneti puffer kezelésével bővített eredeti funkciókat (00, 01, 02, 03-as funkciók, ezekkel beállíthatjuk az adatátvitel jellemzőit, adatot küldhetünk és vehetünk, lekérdezhetjük a státust), valamint az extra szolgáltatásait is használhatjuk. Ezek az eredeti ROM BIOS-ban nem használt 04 és 05-ös funkciókóddal hívhatók. Segítségükkel lehet 19 200 és 38 400 baudos sebességgel is kommunikálni, le lehet tiltani a várakozást a vezérlőjelekre (CTS, DSR) és be lehet állítani az adatáramlás szabályozását kétféle módon: programozottan az XON/XOFF protokollal, hardveresen vezérlőjelek felhasználásával.

Turbo-terminál

Miután a kommunikáció feltételeit ez a program biztosította, el kellett készíteni egy megfelelően gyors terminálprogramot. Ehhez alkalmasnak látszott a Turbo Pascal 5.0 programozási nyelv.

A Turbóban a megszakításhívásokat egyszerűen lehet programozni, így gyorsan elkészültek a kommunikációt támogató eljárások (Get, Put, Init, Status). Maga a főciklus a gyorsaság érdekében igen rövid. Sikerült elérni, hogy 9600 baudnál folyamatos adatáramlás esetén — egy végtelen ciklussal — sem veszt el egyetlen karaktert sem. Többperces folyamatos adatáramlás után 8 MHz-es Turbo XT-nél már bekövetkezhet adatvesztés, de ezt a puffer méretének növelésével (akár 64 kbájtra) meg lehet szüntetni.

A program zömét az egyéb szolgáltatások teszik ki. Ezeket a megfelelő funkcióbillentyűkkel lehet aktivizálni. Segítségükkel egyszerűbb DOS-műveleteket végezhetünk (Directory, Change Directory, Delete), vagy bonyolultabb feladatok esetén kiléphetünk a DOS-hoz, ahonnan egy EXIT utasítással térhetünk vissza a terminálhoz. A program meghívásakor a funkcióbillentyűk szerepéről listát ad; az egyes funkciók az alábbiak.

F1: Dir — a DOS dir parancsát megvalósító funkció, a „Dir mask:" kérdésre a szükséges directory-maszkot kell beírni.

F2: Del — a DOS del (erase) parancsát megvalósító funkció, a megjelenő „Delete filename:" kérdésre a törölni kívánt fájl nevét kell beírni.

F3: Chdir — a DOS chdir (cd) parancsát megvalósító funkció, az „Input new directory:" kérdésre a kívánt útvonalat kell beírni.

F4: Load — a funkció segítségével Intel hexa formátumú fájlt lehet áttölteni a fejlesztendő rendszer RAM-jába. A megjelenő „Enter HEX filename:" kérdésre a fájl nevét kell beírni (********.HEX). A fájl betöltődik a rekordokban adott címinformációknak megfelelő memóriaterületekre. Az áttöltés alatt a képernyőn a memóriacímek jelennek meg, az áttöltés végén hangjelzést ad a program. A fájl nevét paraméterként is átadhatjuk, ekkor az F4 hatására automatikusan a megadott fájlt tölti.

F5: Shell — a megjelenő „Enter DOS command:" kérdésre beírt parancsot végrehajtja, vagy Entert ütve ideiglenesen kiléphetünk a TERMINAL programból a DOS-ba. Az EXIT parancs beírásával térhetünk vissza a TERMINAL programba.

F10: Exit — kilépés kezdeményezése a TERMINAL programból. A képernyőn megjelenő „Are you sure..." kérdésre Y válasz esetén a DOS-ba térünk vissza, N válaszra a program ignorálja a kilépési kérést.

A Turbo Pascal 5.0 a 11 kbájtos szövegfájlból egy 15 kbájtos .EXE fájlt készített, ellentétben a QuickBasickel, ami a hasonló BASIC programból 50 k-t remekelt.

A program forrásnyelvi listája az EXCOM programmal együtt a rovatban közölt címen megrendelhető.

Az EXCOM program

Az IBM PC-ben lévő RS232 típusú soros vonali meghajtót a ROM BIOS-ban lévő program kezeli. Ez a program nagyon kevés szolgáltatást nyújt, ami a vonal kezelését lassúvá és nehézkessé teszi. Ezeket a hiányosságokat hivatott kiküszöbölni az EXCOM program.

A ROM BIOS soros vonalat kezelő eredeti funkciói a következők: adat beolvasása a vonalon; adat kiadása a vonalra; a soros (8250-es típusú) perifériaáramkör felprogramozása; a státusz lekérdezése.

Bővítések

- Vételnél egy 256 bájtos (64 kbájtra bővíthető!) átmeneti puffertár
- Átvitelvezérlés XON/XOFF protokoll szerint (ha az átmeneti tár közel megtelt, akkor egy ^S karaktert küld a vonalra, megállítva az adatküldést. Ha ezek után olvasunk az átmeneti tárból, és csökken a telített állapot, akkor egy ^Q karaktert küld ki, parancsot adva az adatküldés folytatására)
- Átvitelvezérlés a modemvezérlő vonalak felhasználásával
- Az eddigi 9600 bit/s helyett 38 400 bit/s-os maximális adatátviteli sebesség
- Az EXCOM teljesen kompatibilis az eredeti soros kezelőprogrammal. Memóriarezidens, de onnan a rendszer újraindítása nélkül törölhető. A soros vonal modemvezérlő jelei nélkül is üzemelhet

Az EXCOM három megszakítást kezel: a 0BH-t, a második soros vonal hardvermegszakítását; a 0CH-t, az első soros vonal hardvermegszakítását és a 14H-t, a soros vonalak kommunikációs megszakítását.

Mivel a 14H megszakítás funkcióinak (AH=0...3) leírása sok helyen megtalálható, ezért csak a bővített funkciókat ismertetjük.

AH=4 esetén használhatók a kiterjesztett parancsok.

Az AL regiszter bitjeinek jelentése:

- 0 — nem ad DTR jelet
- 1 — nem ad RTS jelet
- 2 — nem ad XON/XOFF protokollt
- 4 — nem vár CTS jelet
- 5 — nem vár DSR jelet
- 6 — 19 200 bit/s
- 7 — 38 400 bit/s

AH = 5 esetén az eredeti ROM BIOS 14H vezérlőprogram állítódik vissza, az EXCOM törlődik a memóriából.

A DTR, RTS, CTS és a DSR a kapcsolatfelvételi jel. Az első kettőt az adapter, a második kettőt a külső egység küldi.

A program az EXCOM.COM program lefuttatásával aktivizálható.

Szinnyei Gerzson

FIGYELEM!

A PÉCÉZZÜNK rovatban megjelent cikkek szövege szövegfájlok formájában, valamint az „Ajándék" szabad szoftver 360 kbájtos DOS-DD lemezen, utánvéttel, önköltségi (lemezár, lemezmásolás, postázás) 300 forintos áron megrendelhető.

***Cím:** Koncz Edit, Budapest, Kunigunda u. 44. 1037*

BÖRZE

TUTTI

ELECTROCOOP
KISSZÖVETKEZET

- Eredeti számítógépek
- Kompatibilis XT/AT, 386 konfigurációk
- Szünetmentes tápegységek
- Egyedi megrendelésfelvétel

Cím: Bp., Üllői út 81. 1091
Tel.: 334-354
Telex: 22-7230
Telefax: 149-869

ENET

Lokális hálózat
IBM kompatibilis gépekre
Külön hardver
nem szükséges
Ára gépenként:
19 900 Ft + áta
ECOSOFT KISSZÖVETKEZET
Tel.: 863-677

COBRA COMPUTER

ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1097 Budapest IX., Illatos út 7. Telefon: 147-6180/388

SZÁMLAKÉSZÍTÉSTŐL A KÖNYVELÉSIG

COBRACONTO

ügyviteli programrendszer moduljai:
- számlakészítő,
- számlanyilvántartó,
- bér- és jövedelemszámfejtő,
- főkönyvi könyvelő,
- anyagnyilvántartó,
- általános könyvelési programok.

KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

ARECO KFT.

a Mikropo KSZ fejlesztési témáinak jogutódja, felajánlja szabad fejlesztőmérnöki kapacitását. ARECO Informatika KFT.
Tel.: 427-453

ASY ELEKTRONIKA

A legújabb ajánlatunkból:
- ASY—16 SZUPERMIKRO számítógépek (12 terminál, UNIX operációs rendszer)
- IBM PC/XT-AT-kompatibilis számítógépek
- megrendelő által definiált betűkészlettel rendelkező billentyűzetek
- elektronikai elemek (integrált áramkörök, ellenállások, tranzisztorok)
- monitordobozok, műszerházak
- szoftvertermékek és fejlesztések

KERESKEDELMI IRODA:
1061 BUDAPEST
LISZT FERENC TÉR 10.
TEL.: 415-166, TELEX: 22-4378

NÁLUNK MÉG KAPHATÓ

Első könyvem a mikróról 30,— Ft
Első könyvem a programról 30,— Ft

CZÜV

Kereskedelmi Iroda
1145 Budapest,
Szugló u. 9—15.
Telefon: 642-000/176-os,
177-es mellék

ISKOLÁKNAK OKTATÓKNAK SZAKKÖRÖKNEK

procontrol

IRÁNYÍTÁSTECHNIKA KISSZÖVETKEZET

MÁRKABOLT, BEMUTATÓTEREM
62/21-165, 28 985
Szeged, Kazinczy u. 8. Tel.: 12-259
Telex: 82-726

386 AT 25 MHz toronygép	299 000 Ft-tól
286 AT 16 MHz TURBO AT gép — 86 XT 12 MHz TURBO XT gép	98 000 Ft-tól, 69 000 Ft-tól,
LAPTOP hordozható LCD XT gép + modem + akku	149 000 Ft
12" zöld monochrom monitor TUV M31	4 900 Ft
5 1/4" DS DD Floppy lemez	79 Ft

és még 2000 árucikkről árjegyzék, azonnali kiszolgálás.

BROTHER AX 15 típusú,
margarétafejes,
elektronikus írógép

Megvásárolható:
Budapest VI.,
Népköztársaság útja 2.
Tel.: 531-231

MIGÉRT

ECONORG

1146 Bp.,
AJTÓSI DÜRER SOR 10.
Levélcím:
1393 Pf.: 319.
Telefon: 421-974
Telex: 22-6544

SZAKÜZLETEK:

1. **Bp. VI., Szinyei Merse u. 1.**
   Tel.:127-628. Tx.: 22-6684
2. **9022 Győr, Lukács S. u. 18.**
   Tel.: 96-14880. Tx.: 02-4679
3. **3100 Salgótarján, Ady Endre u. 1.**
   Tel.: 32-10971. Tx.: 22-9380

Szabványos nyomtatás

ÚTMUTATÓ

I. NYOMTATÓMÓD

Funkció		Rövidítés	Kód
Nyújtott (kétszeres)		SO	14
	vagy	ESC/SO	27,14
Nyújtott ki		DC4	20
Nyújtott be/ki	(n=1 be, n=0 ki)	ESC/W/n	27,87,n
Zsugorított		SI	15
	vagy	ESC/SI	27,15
Zsugorított ki		DC2	18
Kurzív		ESC/4	27,52
Kurzív ki		ESC/5	27,53
Master Print mód	(n lásd lent)	ESC/!/n	27,33,n
Aláhúzás be/ki	(n=1 be, n=0 ki)	ESC/-/n	27,45,n
Kövér		ESC/E	27,69
Kövér ki		ESC/F	27,70
Kettős ráütés		ESC/G	27,71
Kettős ráütés ki		ESC/H	27,72
Elite		ESC/M	27,77
Pica (normál)		ESC/P	27,80
Index	(n=0 felső, n=1 alsó)	ESC/S/n	27,83,n
Index ki		ESC/T	27,84
Proporcionális	(n=1 be, n=0 ki)	ESC/p/n	27,112,n
Betűköz-beállítás	(n=0-127)	ESC/SP/n	27,32,n
Levélminőség (NLQ)	(n=1 be, n=0 ki)	ESC/x/n	27,120,n

Master Print Mode:	Bit	Funkció
	0	(1) Elite
	1	(2) Proporcionális
	2	(4) Zsugorított
	3	(8) Kövér
	4	(16) Kétszeres ráütés
	5	(32) Kétszeres szélesség
	6	(64) Kurzív
	7	(128) Aláhúzás
		(0) Master Print ki

II. GRAFIKUS MÓDOK

Funkció	Rövidítés	Kód
8 tűs grafika	ESC/*/n/l/h	27,42,n,l,h
n=mód 1,h=a grafikus bájtok száma (alacsony - magas bájt sorrendben)		
Sűrűség / Pont/inch		
n=0 egyszeres 60		
1 kétszeres 120		
2 kétszeres, kétszeres sebesség 120		
3 négyszeres 240		
4 képernyő 80		
5 plotter 72		
6 képernyő II. 90		
7 plotter II. 144		
8 pont, egyszeres sűrűség	ESC/K/l/h	27,75,l,h
8 pont, kétszeres sűrűség	ESC/L/l/h	27,76,l,h
8 pont, kétszeres sűrűség kétszeres sebesség	ESC/Y/l/h	27,89,l,h
8 pont, négyszeres sűrűség l,h=a grafikus bájtok száma	ESC/Z/l/h	27,90,l,h
A K,L,Y,Z parancsok cseréje n=lásd ESC/*, m=K,L,Y,Z	ESC/?/n/m	27,90,n,m
9 tűs grafika Kétszer annyi adat szükséges, minden második bájt nyolcadik bitje adja meg, aktív -e a legalsó tű.	ESC/↑/n/l/h	27,94,n,l,h

III. SORTÁVOLSÁG

Funkció		Rövidítés	Kód
Sortávolság 1/8 inch		ESC/0	27,48
7/72 inch		ESC/1	27,49
1/6 inch		ESC/2	27,50
n/216 inch	(n=0-255)	ESC/3/n	27,51,n
n/72 inch	(n=0-85)	ESC/A/n	27,65,n

IV. PAPÍRTOVÁBBÍTÁS

Funkció		Rövidítés	Kód
Soremelés		LF	10
n/216 inch soremelés		ESC/J/n	27,74,n
n/216 inch visszahúzása		ESC/j/n	27,106,n
Automatikus soremelés	(n=1 be, n=0 ki)	ESC/5/n	27,53,n
Lapadagoló be/ki	(n=1 be, n=0 ki)	ESC/EM/n	27,25,n

Előző számunk „Nyomtatók apróban" című írása már foglalkozott az Epson Standard Code for Printers szabvánnyal. Ehhez kapcsolódnak most táblázataink, amelyek az ESC/P nyomtatóparancsokat mutatják be. A parancsnak megfelelő kódok hatására nem karakterek nyomtatódnak, hanem az azt követő karakterek nyomtatásában lesz változás. Az első hasábban az egyes nyomtatófunkciók szöveges leírása található, ezt követi a funkció kód(sorozat)ának megadása az ASCII rövidítések szerint, per jellel (/) tagolva, majd a harmadik hasábban a kódok decimális alakban, vesszővel elválasztva. Itt a nem konkrét értékek (n, x, y stb.) a parancsnak paraméterként részei, kötelezően választhatóak a kívánt hatás elérésére.

Minden több bájtból álló parancs az ESCAPE (27) kóddal kezdődik, ezt követi a tényleges

parancsbájt, végül az esetleges paraméter(ek). Ha a paraméter 0 vagy 1 — általában a valamilyen funkciót ki/be kapcsoló parancsoknál —, akkor megadható akár bináris 0 vagy 1 alakban, akár a „nulla" vagy „egy" karakter ASCII kódjával is (48, 49).

Három példa:

Dőlt betűk írása: ESC/4 decimális .alakban 27,52. E sorozat után dőlt betűket kapunk.

NLQ írás kikapcsolása: ESC/x/0 azaz 27, 120, 48.

81/216 inch soremelés beállítása: ESC/3/Q — 27, 51, 81. Ezután minden soremelés 81/216 inchcsel mozdítja el a papírt. (A „Q" itt paraméter, a 81 kódnak megfelelő karakter!)

Ha egy parancs nem vagy másképpen működik, minden esetben a gépkönyv az irányadó. Egyes nyomtatók ugyanis csak szűkített parancskészlettel rendelkeznek, mások pedig részben eltérnek a szabványtól.

Zoltai Péter

V. FORMÁTUMVEZÉRLÉS

Funkció	Rövidítés	Kód
Vízszintes tabuláció	HT	9
Függőleges tabuláció	VT	11
Lapdobás	FF	12
Függőleges tabulátorcsatorna választása (n=0-7)	ESC///n	27,47,n
Függőleges tabulátor (0. csatorna) beállítása	ESC/B/x/y/z/../0	27,66,x,y,z,..,0
(Legfeljebb 16 függőleges tabulátorpozíciót lehet beállítani, az x,y,z értékeknek növekvő sorrendben kell következniük, a parancs lezárása: '0'		
Függőleges tabulátor (0. csatorna) törlése	ESC/B/0	27,66,0
Függőleges tabulátor (n. csatorna) beállítása (n=0-7)		
A szintaxis azonos az ESC/B-vel!		
Tabulátortörlés (reset)	ESC/R	27,82
Laphosszúság sorokban (n=1-127)	ESC/C/n	27,67,n
Laphosszúság inchben (n=1-22)	ESC/C/0/n	27,67,48,n
Alsó szegély sorokban (n=1-127)	ESC/N/n	27,78,n
Alsó szegély törlése	ESC/O	27,79
Bal margó Pica egységekben (n=0-160)	ESC/l/n	27,108,n
Jobb margó (Pica) (n=1-255)	ESC/Q/n	27,81,n
Bal és jobb margó (Pica)	ESC/X/b/j	27,88,b,j
(b=0-255 j=0-255)		
(Ez a parancs törli a puffert is!)		

VII. KARAKTERKÉSZLET

Funkció	Rövidítés	Kód
Nemzeti karakterkészlet (n=0-10)	ESC/R/n	27,82,n
0 - USA 6 - olasz		
1 - francia 7 - spanyol		
2 - német 8 - japán		
3 - angol 9 - norvég		
4 - dán I. 10 - dán II.		
5 - svéd		
ROM-készlet RAM-ba másolása	ESC/:/0/0/0	27,58,0,0,0
Készletválasztás (n=0 ROM, n=1 RAM)	ESC/%/n/0	27,37,n,0
Karakterdefiniálás	ESC/&/0/a/b/c/...	
a = az első karakter kódja	27,38,0,a,b,c, ..adatbájtok	
b = az utolsó karakter kódja		
Csak egy karakter definiálása: a=b		
c = szélesség proporcionális íráskor		
+11 adatbájt karakterenként (bitkép)		
(a 'c'+11=12 bájt ismétlődik annyiszor, ahány karaktert definiálunk)		

VI. ADATMÓDOSÍTÁS

Funkció	Rövidítés	Kód
Puffer törlése	CAN	24
A puffer utolsó bájtjának törlése	DEL	127
Vezérlőkód-nyomtatás (n=1 be, n=0 ki)	ESC/I/n	27,73,n
A 0-32 kódok esetén karakterek nyomtatódnak		
Vezérlőkód-nyomtatás II.	ESC/6	27,54
U.a. a 128-159 kódokra		
Vezérlőkód nyomtatás II. kikapcsolása	ESC/7	27,55
Nyolcadik bit '0'-ra állítása	ESC/=	27,61
Nyolcadik bit '1'-re állítása	ESC/>	27,62
Bitbeállítás kikapcsolása	ESC/#	27,35

VIII. EGYÉB KÓDOK

Funkció	Rövidítés	Kód
Csengő	BEL	7
Kocsi-vissza	CR	13
Papírvég-érzékelő kikapcsolása	ESC/8	27,56
bekapcsolása	ESC/9	27,57
Egy sor egyirányú nyomtatása	ESC/<	27,60
Egyirányú nyomtatás (n=1 be, n=0 ki)	ESC/U/n	27,86,n
Írógépmód (n=1 be, n=0 ki)	ESC/i/n	27,105,n
Felsebesség, csendes uzem (n= -//-)	ESC/s/n	27,115,n
Nyomtató OFF-LINE	DC3	19
ON-LINE	DC1	17
Nyomtató RESET	ESC/§	27,64

Az AmigaBasic utasításkészlete

OBJECT.START [object-szám1 [,object-szám2...]]
OBJECT.STOP [object-szám1 [,object-szám2...]]
Engedélyezi, illetve tiltja az objectek mozgását.
OBJECT.VX object-szám, sebesség
OBJECT.VY object-szám, sebesség
Az object sebességét állítja be a megadott (x vagy y) irányban. Mértékegység: képpont/szekundum. Tartomány: —32768...+32767.
y=OBJECT.VX(object-szám)
y=OBJECT.VY(object-szám)
Az object sebességét visszaadó függvény. Mértékegység: képpont/szekundum.
OBJECT.X object-szám,x
OBJECT.Y object-szám,y
Az object bal felső sarkának koordinátáit adhatjuk meg ezzel az utasítással.
y=OBJECT.X(object-szám)
y=OBJECT.Y(object-szám)
A függvény az object bal felső sarkának koordinátáit szolgáltatja.
y$=OCT$(x)
A függvény x oktális (8-as számrendszerbeli) értékét adja meg.
ON BREAK GOSUB címke
Programmegszakítási kísérlet esetén a címkével megjelölt sorra adódik a vezérlés.
ON COLLISION GOSUB címke
Objectek ütközése esetén a megadott szubrutinra történő ugrást hajt végre az AmigaBasic.
ON ERROR GOTO címke
Programhiba bekövetkeztével a címkével jelölt sorra adódik a vezérlés.
ON kifejezés GOSUB címke1 [,címke2...]
ON kifejezés GOTO címke1 [,címke2...]
A vezérlésátadás mindkét utasításnál azzal a címkével jelölt sorra történik, amelynek a helye a címkelistában a kifejezés értékével egyenlő.
ON MENU GOSUB címke
Bármely menütétel (menüpont) kiválasztása esetén az AmigaBasic meghívja a címkéhez tartozó szubrutint.
ON MOUSE GOSUB címke
Az egér bal billentyűjének aktivizálása esetén a megadott szubrutinra adódik a vezérlés.
ON TIMER (másodperc) GOSUB címke
A másodpercekben beállított időtartam leteltével a rendszeróra megszakítást generál, amelynek hatására a címkével jelölt programsorra adódik a vezérlés.
OPEN fájlnév [FOR üzemmód1] AS [#] logikai fájlszám [LEN=rekordhossz]
OPEN üzemmód2, [#] logikai fájlszám, fájlnév [,rekordhossz]
Megnyitja a megadott fájlt, amelyre azután a logikai fájl számával hivatkozhatunk. A rekordhosszal megszabható, hogy két egymást követő hozzáférés kezdőpontja milyen távol legyen egymástól. Üzemmód1 lehetséges változatai:
OUTPUT: soros kivitel
APPEND: már létező fájl bővítése
INPUT: soros behozatal
Üzemmód2 helyére a következő, *idézőjelek közé tett* karakterek helyettesíthetők be:
O: soros kivitel
A: már létező fájl bővítése
I: soros behozatal
R: soros behozatal/kivitel
OPEN „COM1: [Baudrate] [,paritás] [,szóhossz] [,stopbitek száma]" [FOR üzemmód] AS [#] logikai fájl száma
Megnyit egy fájlt az RS232-es porton. A Baudrate (bitsebesség) szabványos értékeket vehet fel: 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, 19 200 bit/s.
A paritás típusválasztéka:
E: páros paritás
O: páratlan paritás
N: nincs paritásképzés
A szóhossz 5, 6, 7 vagy 8 bit, a stopbitek száma pedig 1 vagy 2 lehet.
Kétféle üzemmód állítható be:
OUTPUT: soros kivitel
INPUT: soros behozatal
OPTION BASE 0/1
Azt szabja meg, hogy a tömbváltozók 0-ás vagy 1-es indexszel kezdődjenek. (Ellenőrzés: LBOUND).
PAINT [STEP] (x,y) [,szín[,kontúrszín]]
Az (x,y) pontból kiindulva, a PATTERN-nel megadott textúrával kitölti a zárt alakzat belsejét. Megadhatjuk azt a színt, amellyel a kitöltést végzi, és azt a kontúrszínt, amelyet határnak tekint e művelet során.
PALETTE színszám, vörös, zöld, kék
Beállítja a képernyőn egyidejűleg megjeleníthető színkészlet — a paletta — egyik tagját, a három színösszetevő megadásával. A színösszetevők 0-tól 1-ig terjedő tizedestörtek. A Workbenchben négy palettaszín használható (0...3): kék (0), fehér (1), fekete (2), narancssárga (3). Érdekességként megemlítem, hogy az egérrel mozgatott nyíl (pointer) a 17...19-es palettaszínekkel van megrajzolva. A PALETTE-tel a teljes (16 ^ 3=4096 színből álló) színskála átfogható.
PATTERN [vonalmaszk] [,kitöltőmaszk]
Az AREAFILL, a PAINT és a LINE utasításoknál megjelenő mintázatot lehet beállítani vele. A vonalmaszk a LINE utasítás vonal- és üres-téglalap-rajzoló üzemmódját befolyásolja. Ez egy 16 bites szám, amelyben az 1 értékű bitek előtérszínt, a 0 értékűek háttérszínt jelentenek. &Hf0f0-ás vonalmaszkkal például szaggatott vonalat húzhatunk. A kitöltőmaszk egy 16 bites, egydimenziós numerikus tömb, elemeinek száma a mintázat függőleges irányú ismétlési közét határozza meg, a vízszintes méret pedig állandó (16 képpont). A kitöltőmaszkban levő 1-es bitek előtér-, a 0-ás bitek háttérszínt jelentenek. A megadáskor a tömbváltozót mint egyszerű numerikus változót kell feltüntetni, tehát indexek nélkül.
y=PEEK(memóriacím)
A memóriacím tartalmát adja meg egybájtos formában.
y=PEEKL(memóriacím)
A memóriacím tartalmát kapjuk meg 32 bites formában.
y=PEEKW(memóriacím)
A memóriacím tartalmát adja vissza 16 bites formában.
y=POINT(x,y)
A képpont színét tudhatjuk meg e függvény segítségével.
POKE memóriacím,n
A 8 bites szám a megadott memóriacímre kerül.
POKEL memóriacím,n
A 32 bites számot beírja a megcímzett memóriahelyre.
POKEW memóriacím,n
A 16 bites számot a megadott memóriacímre írja.
y=POS(ablakszám)
A kijelölt ablak szöveges kurzorának oszloppozícióját visszaadó függvény.
PRESET [STEP] (x,y) [,szín]
PSET [STEP] (x,y) [,szín]
A képpontot a megadott színűre állítja be. Ha nem adunk meg színt, a PSET az 1-es palettaszínt, a PRESET pedig a 0-ás palettaszínt használja.
PRINT [kifejezések sora]
A kifejezések kiíratása.
PRINT USING u$; kifejezések sora [;]
A kifejezések formázott kiíratására szolgál. u$ az ehhez szükséges maszkot tartalmazza. A maszkban alkalmazható speciális karakterek:
!: csak az első karaktert írja ki

&: minden karaktert kiír
#: számjegy-szimbólum
+: pozitív szám esetén kiírja a „+" jelet, méghozzá azon a helyen, ahol a maszkban áll
—: a „—" jelet ott írja ki, ahol az a maszkban áll
*: az üres karakterhelyeket *-gal tölti fel
^^^^: normál alakban írja ki a számot
_: azt jelzi, hogy a maszkban ezután következő karaktert változtatás nélkül kell kijelezni
Egy megjegyzés: ha a szám egész része több jegyből áll, mint amit a maszk (#) megenged, akkor a kiíráskor egy „%" jel figyelmeztet erre a hibára.

PRINT #logikai fájlszám, [USING u$;] kifejezések listája
Fájlok formázott írására szolgáló utasítás. (Lásd: PRINT USING)

PRINT PTAB(n)
Használatával képpontban adhatjuk meg a tabulálás mértékét. Tartomány: 0...32767.

PUT [#] logikai fájlszám [,rekordszám]
A buffer tartalmát kiírja a fájl soron következő vagy egy tetszőleges rekordjába.

PUT [STEP] (x,y),tömbváltozó [(index1 [,index2...])],üzemmód
Megjeleníti a GET-tel elmentett képrészletet úgy, hogy annak bal felső sarka az (x,y) pontba essen. A lehetséges üzemmódok:
PSET: normál módú megjelenítés
PRESET: a színek invertálódnak
AND: a lerakott grafika és a háttér között pontról pontra ÉS kapcsolatot képez
OR: a lerakott grafika és a háttér között pontról pontra VAGY kapcsolatot képez
XOR: a lerakott grafika és a háttér között pontról pontra KIZÁRÓ VAGY kapcsolatot képez

RANDOMIZE [kezdőérték]
RANDOMIZE TIMER
Új kezdőértékről indítja a véletlenszám-generátort. A második szintaxist alkalmazva a kezdőérték az időzítő pillanatnyi állása lesz.

READ változó1 [,változó2...]
Sorban feltölti a változókat a DATA listaelemekkel. (Lásd: DATA)

REM [megjegyzés]
A program működését befolyásoló megjegyzést szúrhatunk be vele a programlistába.

RESTORE [címke]
Kiválasztja a következő READ utasítással olvasni kívánt DATA listaelemet. Ha nem adunk meg címkét, akkor azt eredményezi, hogy a READ utasítás a legelső DATA után álló legelső értéket fogja beolvasni. Címkét megadva pedig a címke utáni első DATA utáni első listaelemet fogja beolvasni a READ.

RESUME [0]
RESUME NEXT
RESUME címke
Az ON ERROR által deklarált hibarutin szabályos befejezésére szolgáló utasítás. Az első szintaxist alkalmazva a hibát okozó utasítást újból végrehajtja, a második esetén a hibás utasítást követő utasítással folytatja a programot, a harmadik alapján pedig a megadott címkére tér vissza.

y$=RIGHT$(x$,n)
Sztringfüggvény, amely az x$ utolsó n darab karakterét adja.

y=RND[(x)]
Egy 0 és 1 közé eső álvéletlen számot generál. Az alábbi értékeket helyettesíthetjük be x helyére:
x > 0 vagy nem is
szerepel: A következő véletlenszám a régiből ered.
x < 0: A véletlenszám-generátor kezdőértékét átveszi az Amiga-Basic.
x=0: Megismétli az utolsó álvéletlen számot.

RSET sztringváltozó=x$
Feltölti a FIELD segítségével valamely fájlhoz hozzárendelt buffert x$ tartalmával. x$-et a bufferben jobbra igazítva helyezi el. Ezután jöhet a PUT # utasítás.

RUN [címke]
RUN programnév[,R]
Törli az összes változót, valamint a képernyőt, majd elindítja a programot a megadott címkétől, vagy pedig betölti azt, és ezután indítja el. Ez utóbbi esetben, ha R opciót is megadunk, akkor az összes addig megnyitott fájl nyitva marad.

y=SADD(sztringkifejezés)
A kifejezés kezdőcímét adja meg a memóriában.

SAVE [programnév] [,A] [,P] [,B]
Elmenti a programot a kívánt néven:
A: ASCII formátumban
P: védett (nem listázható) formában
B: bináris formában

SAY sztringváltozó [,üzemmód]
Kimondja a szringváltozóban elhelyezett fonémakódoknak megfelelő fonémákat. Az üzemmód 9 paramétert foglal magába:
0: Alapfrekvencia Hz-ben (110 Hz=normál férfi hang). Tartomány: 65...320.
1: Moduláció. 0: szótagmoduláció és hangsúlyozás, 1: monoton beszéd (robothang).
2: Sebesség szó/percben (alapbeállítás=150). Tartomány: 40...400.
3: A beszélő neme. 0: férfi (alapbeállítás); 1: nő.
4: Hangszín (alapbeállítás=22 200). Tartomány: 5000—28 000.
5: Hangerő (alapbeállítás=64). Tartomány: 0...64.
6: Hangcsatorna:
0: 0-ás csatorna
1: 1-es csatorna
2: 2-es csatorna
3: 3-as csatorna
4: 0-ás és 1-es csatorna
5: 0-ás és 2-es csatorna
6: 1-es és 3-as csatorna
7: 2-es és 3-as csatorna
8: minden szabad bal csatorna
9: minden szabad jobb csatorna
10: minden szabad csatornapár
11: minden szabad csatornapár
7: Szinkronizáció:
0: végigmondja és csak utána lép a következő utasításra (szinkron üzemmód)
1: elkezdi kimondani, de közben a következő utasításra lép (aszinkron üzemmód)
8: A 7-es üzemmód-paraméter 1-es értéke esetén:
0: befejezi az aktuális SAY utasítás végrehajtását, mielőtt egy új SAY-t elkezdene
1: nem hajtja végre a SAY-t
2: azonnal abbahagyja az aktuális SAY-t, és elkezdi az újat
A paramétereket úgy adhatjuk meg, hogy egy 16 bites numerikus tömbbe írjuk be őket, és a tömbváltozó nevét írjuk le a SAY utasítás után.

SCREEN szám, szélesség, magasság, bitsíkok száma, felbontás
SCREEN CLOSE szám
Új képernyőt (sreent) nyit, illetve már létező képernyőt megszüntet (CLOSE). Egyszerre legfeljebb négy képernyő lehet nyitva, amelyek az 1-től 4-ig terjedő számokat viselhetik. A szélesség és a magasság paramétereknek kijelzési korlátai nincsenek, csupán a rendelkezésre álló memóriakapacitás szabja meg a felső határt. A következő felbontásértékek állíthatók be:
1: 320×256 (LORES, NON INTERLACE)
2: 640×256 (HIRES, NON INTERLACE)
3: 320×512 (LORES, INTERLACE)
4: 640×512 (HIRES, INTERLACE)
A bitsíkok száma a paletta színeinek számát határozza meg:
1: 2 színű üzemmód
2: 4 színű üzemmód
3: 8 színű üzemmód
4: 16 színű üzemmód
5: 32 színű üzemmód (HIRES ÜZEMMÓDBAN NEM LÉTEZIK)

SCROLL (x1,y1)—(x2,y2),dx,dy
Eltolja a kijelölt téglalap által határolt grafikát a dx-nek és dy-nak megfelelő irányban. (x1,y1) a téglalap bal felső, (x2,y2) a téglalap jobb alsó sarkának koordinátái. Az eltolás egysége 1 képpont. Ha az eltolás negatív, akkor balra vagy felfelé, ha pozitív, akkor jobbra vagy lefelé tolódik el a rajzolat.

y=SGN(x)
Szignumfüggvény. Értéke x < 0 esetén —1, x=0 esetén 0, x > 0 esetén pedig +1.

SHARED változó1 [,változó2...]
Ezt az utasítást csak alprogramban (SUB) szabad használni, és hatására a felsorolt változók tartalma a főprogram számára is elérhetővé válik. A felsorolásban a tömbváltozók neve után üres zárójelet kell tenni.

y=SIN(x)
Szinuszfüggvény

SLEPP
Várakozás bármilyen felhasználói aktivitásra, megszakításra. A

programfutás mindaddig felfüggesztődik, amíg az alábbi események közül valamelyik be nem következik:
— az egér bal billentyűjének megnyomása,
— bármelyik billentyű megnyomása,
— objectek ütközése,
— egy menüpont kiválasztása,
— az időzítő (timer) megszakításkérése.

SOUND frekvencia, időtartam [,[hangerő] [,csatorna]]
SOUND WAIT
SOUND RESUME
Kiadja a beállított hangot, a WAVE utasítással definiált hullámformát használva. A frekvenciát Hz-ben kell érteni, és 20-tól 15 000 Hz-ig adható meg. Az időtartam 0-tól 77-ig, a hangerő 0-tól 64-ig értelmezett. 1 másodperces időtartamnak 18.2-es érték felel meg. A csatorna (0...3) kiválasztása egyértelműen megmondja a hullámformát is, mert azt a WAVE utasítás az egyes csatornaszámokhoz rendeli hozzá. A második és a harmadik szintaxis a csatornák szinkronizálását oldja meg. Többszólamú zene esetén ki kell adni egy SOUND WAIT utasítást, majd fel kell programozni az egyes csatornákat az első szintaxis szerint. Ezután egy SOUND RESUME-ot kiadva egyszerre szólal meg az összes hang.

y$=SPACE$(n)
A függvény n darab (max. 32 767) szóközből álló karakterláncot hoz létre.

SPC(n)
n darab (max. 255) szóközt adó függvény, amely csak a PRINT, PRINT # és a LPRINT utasításokkal együtt használható.

y=SQR(x)
Négyzetgyökfüggvény.

y=STICK(n)
A botkormány állapotát lekérdező függvény. n lehetséges értékei az alábbiak:
0: joystick 1, x irány
1: joystick 1, y irány
2: joystick 2, x irány
3: joystick 2, y irány
A függvénytől kapott értékek jelentése:
1: fel vagy jobbra
0: nincs kimozdítva
—1: le vagy balra

STOP
Megállítja a programfutást, amely ezután CONT-tal folytatható.

y$=STR$(x)
A numerikus argumentumból sztringet képez.

y=STRING(n)
A botkormány tűzgombjának lekérdezésére szolgáló függvény. n értéke lehet:
0: —1-et ad, ha az 1-es botkormány tűzgombja az utolsó STRING (Ø) lekérdezés óta le lett nyomva
1: —1-et ad, ha az 1-es botkormány tűzgombja le van nyomva
2: mint az n=0, csak a 2-es botkormányra
3: mint az n=1, csak a 2-es botkormányra

y$=STRING$(n,karakterkód)
y$=STRING$(n,x$)
n hosszúságú, a megadott karakterrel feltöltött karakterláncot hoz létre.

SUB alprogram-név [(változók felsorolása)] STATIC
END SUB
EXIT SUB
Az első utasítással egy alprogram kezdetét jelölhetjük. Az alprogram a felsorolt változókban veszi át a CALL utasítással átadott argumentumlistát. Az alprogram végét egy END SUB jelzi, amely utasítást ad a főprogramba való visszatérésre. Az EXIT SUB kiadásával bármikor kiléphetünk az alprogramból. Megjegyzés: a tömbváltozók neve után üres zárójelet kell írni!

SWAP változó1, változó2.
Felcseréli a két változó tartalmát.

SYSTEM
Kilépés az AmigaBasicből.

TAB(n)
A kurzortól számított n karaktert (max. 255) üresen hagyja (tabulálja).

y=TAN(x)
Tangensfüggvény.

y$=TIME$
A rendszeridőt tartalmazza ez a sztringváltozó „óra:perc:másodperc" formátumban.

TIMER ON/OFF/STOP
A rendszeróra által generált megszakítást (ON TIMER GOSUB) engedélyezi, illetve tiltja.
ON: engedélyezi az ON TIMER GOSUB működését
OFF: kikapcsolja az ON TIMER GOSUB-ot és az időzítő megszakításait
STOP: kikapcsolja az ON TIMER GOSUB-ot, de engedélyezi a megszakítások eltárolását

y=TIMER
Az időzítő (timer) állítását adja meg másodpercekben.

y$=TRANSLATE$(szöveg)
Előállítja a szövegből az ún. fonémakód-sztringet, amelyet azután a SAY utasítás képes kimondani.

TORN
TROFF
Be-, illetve kikapcsolja a nyomkövető (trace) üzemmódot.

y$=UCASE$(x$)
A tetszőleges karakterekből álló karakterlánc kisbetűit nagybetűkre cseréli ki.

y=VAL(x$)
A sztringet numerikus formára alakítja át.

y=VARPTR(változó)
A változó kezdőcímét adja meg a memóriában.

WAVE csatornaszám, definíció
A hullámforma definícióját rendeli hozzá a megadott csatornához. A definíció egy egydimenziós, 256 elemű, 16 bites számtömb, amelyet fel kell tölteni a hullámformát pontról pontra leíró értékekkel. A tömb elemei a —128...+127-es tartományba eshetnek. A definíció helyére csak a tömbváltozó nevét kell beírni. Létezik egy könnyítés szinuszos jelalak esetén, ilyenkor a definíció helyére a SIN kulcsszót kell írni.

WHILE kifejezés
.
.
.
WEND
Ciklusszervező utasításpár. A FOR-NEXT párostól leginkább abban különbözik, hogy a feltételvizsgálat a ciklus fejénél (WHILE) történik. Ha a kifejezés értéke hamis, a WEND utáni sorra kerül a vezérlés, egyébként pedig újból végrehajtódik a ciklusmag.

WIDTH [LPRINT] [szélesség] [,tabulátorhossz]
WIDTH # logikai fájlszám [,szélesség] [,tabulátorhossz]
WIDTH készülék [,szélesség] [,tabulátorhossz]
Beállítja a sorok hosszát (szélesség) és a tabulátorpozíciók távolságát (tabulátorhossz) a nyomtatón, a képernyőn és az RS232-es csatornán. A készülékekről (logikai eszközökről) a bevezetőben volt szó.

WINDOW szám [,[név] [,[(x1,y1,)—(x2,y2)] [,típus] [,screen-szám]]]]
WINDOW OUTPUT szám
WINDOW CLOSE szám
Az első szintaxis új ablakot nyit a megadott képernyőn, a második az aktív kiviteli ablakot állítja be, a harmadik pedig megszüntet egy már létező ablakot. Egy ablakra később annak számával hivatkozhatunk, és nevet is adhatunk neki, ami meg fog jelenni a fejlécen. Az (x1,y1) a bal felső sarok, az (x2,y2) a jobb alsó sarok koordinátái az illető képernyő koordinátarendszerében. Vigyázat! Az ablak belső, felhasználható méretei kisebbek a megnyitáskor megadott méreteknél! Az effektív magasság ennél 14 képponttal, az effektív szélesség pedig 9 képponttal kisebb. A típust a következő értékek összeadásával képezhetjük ki:
1: az ablak nagysága megváltoztatható
2: az ablak elmozgatható
4: az ablak előtérbe és háttérbe helyezhető
8: az ablak kikapcsolható
16: az ablak tartalma újból előtűnik, ha egy őt takaró másik ablakot levesznek róla
Figyelem! A 17-es érték igen memóriaigényes!

y=WINDOW(n)
Információkat ad az aktív kiviteli ablakról. n a következő értékeket veheti fel:
0: az egérrel kiválasztott ablak számát adja meg
1: az aktív kiviteli ablak számát adja vissza
2: az aktív kiviteli ablak szélessége
3: az aktív kiviteli ablak magassága
4: az aktív kiviteli ablak szöveges kurzorának x koordinátaja
5: az aktív kiviteli ablak szöveges kurzorának y koordinátája
6: az aktív kiviteli ablakban használható színek száma
7: az aktív kiviteli ablakot leíró adatstruktúra kezdőcímére mutató vektort adja vissza
8: az aktív kiviteli ablakhoz tartozó ún. Rastport struktúra kezdőcímét adja meg

WRITE [kifejezések felsorolása]
Lényegét tekintve megegyezik a PRINT-tel, de a WRITE az összes utána álló karaktert megjeleníti (idézőjelet, pontosvesszőt stb. is).

WRITE [#] logikai fájlszám, kifejezések felsorolása
A PRINT # megfelelője, de az összes karaktert kiviszi a fájlba (idézőjel, vessző stb).

Tóth Zoltán

Amire jó a PIA

Az Atari számítógépek Motorola 6820-as processzorra épülő PIA áramköre (lásd az *ábrát*) 2 darab 8 bites párhuzamos portot tartalmaz (PAO—7, PBO—7), az adatátvitelt a CA1, CA2, CB1, CB2 kontrollbitek segítik elő. Az utóbbi négy bit univerzális, az áramkör beprogramozásától függően többféleképpen használható. A PIA két része — A és B — egymástól függetlenül használható, s a két rész bizonyos minimális eltérésektől eltekintve ekvivalensnek mondható.

Az A port 8 adatbitje (PAO—7) a botkormány-illesztőkre van kivezetve, a B port néhány bitje az MMU-t (Memory Management Unit) irányítja, a kontrollbitek pedig — nem hagyományos módon — a soros adatátvitelnél segítenek. A PIA egyes kivezetéseinek csatlakoztatását a *táblázat* mutatja.

Az A, illetve B portok bitjei egyenként állíthatók ki- vagy bemenetre. Ezt az adatirány-regiszterek teszik lehetővé (DDRA, DDRB — Data Direction Register). Ha valamely port adatirány-regiszterének egy bitjét log.1-re állítottuk, akkor az illető port megfelelő bitje kimenet lesz, ellenkező esetben bemenet.

Az adatirány-regisztereken kívül még négy, programból elérhető regisztert tartalmaz a PIA: két adatregisztert (DRA, DRB — Data Register) és két kontrollregisztert (PORTACNTL, PORTBCNTL). A kontrollregiszterbe írt parancsbájttal lehet a PIA-nak többek között azt is előírni, hogy a továbbiakban az illető portra küldött adatok a DR-be vagy a DDR-be kerüljenek.

Úgy tervezték a PIA áramkört, hogy kézfogásos adatátvitel esetén (handshaking) az A port kontrollbitjei az A portot mint bemeneti portot, a B port kontrollbitjei pedig a B portot mint kimeneti portot támogatják. Ebből adódik a CA2, illetve a CB2 eltérő működése.

A PORTACNTL kontrollregiszter bitjeit a következő funkciók jellemzik:

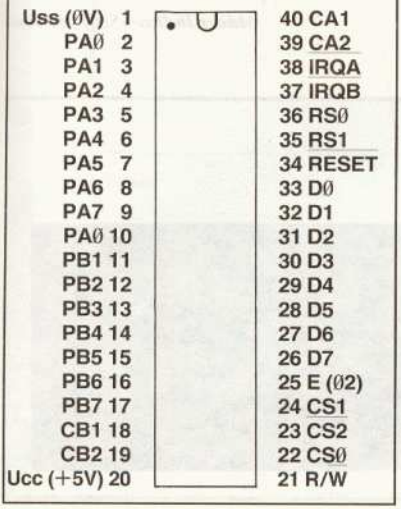
Bit	Funkció
0,1	A CA1 irányítása
2	Ha értéke 0, akkor a továbbiakban az A porton keresztül a DDRA-t lehet elérni, ellenkező esetben a DRA-t
3—5	A CA2 irányítása
6	IRQA1. A PORTACNTL-t olvasva értéke log.1 lesz, ha korábban a CA2-re megszakításjel érkezett. Az A port olvasásakor automatikusan kinullázódik
7	IRQA2. Funkciója olyan, mint a 6. bité, de ez a CA1-re érkező megszakításjel hatására állítódik log.1-re.

A CA1 irányítása a következő elvet követi:

1. bit	0. bit	Értelmezés
0	0	Ha a CA1-en 1—0 átmenet van, a PORTACNTL 7. bitje log.1-re állítódik, a PIA $\overline{\text{IRQA}}$ kimenete log.1-en marad
0	1	Ha a CA1-en 1—0 átmenet van, a PORTACNTL 7. bitje log.1-re állítódik, a PIA $\overline{\text{IRQA}}$ kimenete pedig log.0-ra esik (megszakítás)
1	0	Ha a CA1-en 0—1 átmenet van, a PORTACNTL 7. bitje log.1-re állítódik, a PIA $\overline{\text{IRQA}}$ kimenete log.1-en marad
1	1	Ha a CA1-en 0—1 átmenet van, a PORTACNTL 7. bitje log.1-re állítódik, a PIA $\overline{\text{IRQA}}$ kimenete pedig log.0-ra esik (megszakítás)

A CA2 esetén mindez természetesen módosul. Ha a PORTACNTL 5. bitje 0, akkor a CA2 ugyanolyan megszakítás-bemenetként fog működni, mint a CA1. A PORTACNTL 4. és 3. bitjének értelmezése ebben az esetben ugyanolyan, mint a CA1 irányításánál az 1. és 0. bitnek, CA1 helyén mindenütt CA2-t, a PORTACNTL 7. bitje helyén pedig mindenütt 6. bitet behelyettesítve.

A PIA lábkiosztása



A PIA kivezetéseinek csatlakoztatása

	PIA-kivezetés	Csatlakoztatás (csatlakozó pin)	
A port	PAØ	JOYSTICK1, előre	1
	PA1	JOYSTICK1, hátra	2
	PA2	JOYSTICK1, balra	3
	PA3	JOYSTICK1, jobbra	4
	PA4	JOYSTICK2, előre	1
	PA5	JOYSTICK2, hátra	2
	PA6	JOYSTICK2, balra	3
	PA7	JOYSTICK2, jobbra	4
	CA1	SIO — „Proceed"	9
	CA2	SIO — „Motor-Control"	8
B port	PBØ	MMU — ROM/RAM az operációs rendszer tartományában	
	PB1	MMU — BASEN...BASIC-ROM bekapcsolása	
	PB2—6	nincs csatlakozás	
	PB7	MMU — TEST...TEST-ROM bekapcsolása	
	CB1	SIO — „Interrupt"	13
	CB2	SIO — „Command"	7

Ha a PORTACNTL 5. bitje 1, akkor a CA2 kimenet lesz.

4. bit	3. bit	Értelmezés
0	0	Az A portról olvasva az órajel 1—0 átmenetével egy időben, a CA2 0-ra eseik. Amikor a PORTACNTL 7. bitje CA1 révén log.1-re állítódik, CA2 visszatér log.1-re
0	1	Az A portról olvasva az órajel 1—0 átmenetével egy időben a CA2 0-ra esik, majd az órajel következő 1—0 átmeneténél visszamegy log.1-re
1	0	Statikus kimenet, log.0
1	1	Statikus kimenet, log.1

A B port programozása nagyon hasonlít az A portéhoz.

A PORTBCNTL bitjeinek értelmezése a következő:

Bit	Funkció
0,1	A CB1 irányítása
2	Ha értéke 0, akkor a továbbiakban a B porton keresztül a DDRB lesz elérhető, ellenkező esetben a DRB
3—5	A CB2 irányítása
6	IRQB1. PORTBCNTL-t olvasva értéke log.1 lesz, ha korábban a CB2-re megszakításjel érkezett. Ha a B portra írunk, az IRQB1 automatikusan kinullázódik
7	IRQB2. Funkciója olyan, mint a 6. bité, de ez a CB1-re érkező megszakításjel hatására állítódik log.1-re

A CB1 irányítása ugyanolyan, mint a CA1-é, de $\overline{\text{IRQA}}$ helyett $\overline{\text{IRQB}}$-t, PORTACNTL helyett pedig PORTBCNTL-t kell érteni.

A CB2 irányítása egy kissé eltér a CA2 irányításától, ami abból adódik, hogy a B portnak handshakingnél kimeneti funkciója van. Ha a PORTBCNTL 5. bitje 0, akkor CB2 megszakításbemenetként kűködik:

4. bit	3. bit	Értelmezés
0	0	Ha a CB2-n 1—0 átmenet van, a PORTBCNTL 6. bitje log.1-re állítódik, a PIA $\overline{\text{IRQB}}$ kimenete log.1-en marad
0	1	Ha a CB2-n 1—0 átmenet van, a PORTBCNTL 6. bitje log.1-re állítódik, az $\overline{\text{IRQB}}$ kimenet pedig log.0-ra esik (megszakítás)
1	0	Ha a CB2-n 0—1 átmenet van, a PORTBCNTL 6. bitje log.1-re állítódik, az $\overline{\text{IRQB}}$ kimenet log.1-en marad
1	1	Ha a CB2-n 0—1 átmenet van, a PORTBCNTL 6. bitje log.1-re állítódik, az $\overline{\text{IRQB}}$ kimenet pedig log.0-ra esik (megszakítás)

Ha a PORTBCNTL 5. bitje 1, akkor a CB2 kimenet lesz.

4. bit	3. bit	Értelmezés
0	0	A B portba írva, az órajel 0—1 átmenetével egy időben CB2 0-ra esik. Amikor a PORTBCNTL 7. bitje CB1 által log.1-re állítódik, CB2 visszamegy log.1-re
0	1	A B portba írva, az órajel 0—1 átmenetével egy időben CB2 0-ra esik, majd az órajel következő 0—1 átmeneténél visszamegy log.1-re
1	0	Statikus kimenet, log.0
1	1	Statikus kimenet, log.1

Ha a fentebb leírtakat gondosan áttanulmányozzuk, láthatjuk, hogy a PIA milyen nagy lehetőségeket nyújt a párhuzamos adatátvitelhez. Képes megszakítani a processzor tevékenységét, ha a PIA-ra kívülről átveendő adat érkezett, vagy ha a perifériás egység — például a nyomtató — átvette a PIA-tól a neki küldött adatot. A megszakítás emellett tiltható, miközben a programból a kontrollregiszterek lekérdezésével megállapítható, teljesült-e a megszakítás feltétele. Engedélyezett megszakítás esetén pedig a megszakításrutin a kontrollregiszterek 6. és 7. bitjét lekérdezve, megállapíthatja a megszakítás okát.

Az Atari 130 XE, 65 XE, 800 XL, 800 XE számítógepekben a PIA — mint a bevezetőben írtuk — nem adatátvitelre szolgál. Ezért óvatosan kell eljárni a PIA programozásánál, mert például az MMU hibás irányításával (B port) könnyen „befagyhat" a gép.

A PIA egyes regiszterei a következő címeken érhetők el:

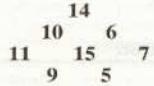
PORTA	54016	$D300
PORTB	54017	$D301
PORTACNTL	54018	$D302
PORTBCNTL	54019	$D303

A korábban leírtak értelmében a PORTA, illetve a PORTB címeken az adatregiszterek vagy az adatirány-regiszterek érhetők el, a konrtollregiszterbe írt parancsbájt alapján.

A másodpercenként 50-szer lefutó VBLANK megszakításrutin minden alkalommal átmásolja az A port tartalmát a botkormány-regiszterekbe, melyek a következő címeken találhatók:

JOYSTICK1	632	$278
JOYSTICK2	633	$279

Ezeket a regisztereket olvasva egyszerűen megállapíthatjuk a botkormány helyzetét az alábbi ábra segítségével:



Máder Indira—Szíjjártó Jenő

FOK GYEM

FOK-GYEM Szövetkezet 1775 Budapest, Pf. 69
Finommechanikai és
Elektronikus
Műszergyártó Szövetkezet
1222 Budapest,
Nagytétényi út 100—102.
Tel.: 173-0011
Telex: 22-60-34

Sporttermi univerzális kisberendezés. Alkalmas kisebb sporttermekben rendezendő sportversenyek időmérő és numerikus közönségtájékoztató feladatainak ellátására.
Bonyolítható sportversenyek:
kosárlabda, kézilabda, röplabda, tenisz, asztalitenisz, birkózás, tollaslabda, súlyemelés, atlétika, torna, labdarúgás, jégkorong, műkorcsolya, vízilabda, vívás, ökölvívás
Megrendelhető a Szövetkezet kereskedelmi főosztályán.

Az értékelőfüggvény hatékonyságának elemzése

Októberi számunkban mutattuk be az értékelőfüggvény végső formáját. Az állás végső pontértékét úgy kapjuk meg, hogy az értékelőfüggvényt a világos és a sötét bábukra egyaránt alkalmazzuk, majd a két érték különbségét képezzük — ezt ezentúl relatív értékelésnek nevezzük.

Érdemes összehasonlítani az értékelőfüggvény képeit azokban az esetekben, amikor az átlagszámításhoz kiválasztott játszmák más-más kritériumnak tesznek eleget.

Az *1. ábrán* az összes feldolgozott játszma átlagát láthatjuk (832 mesterjátszma).

A *2. ábrán* csak azoknak a játszmáknak képeztük az átlagát, amelyek negyven lépésen belül befejeződtek. A feldolgozott játszmák huszonnyolc százaléka tartozik ebbe a kategóriába, szám szerint 234.

A *3. ábra* esetében csak azokat a játszmákat vettük figyelembe, amelyek harminc lépésen belül eldőltek. Ez összesen 69 játszmát jelent, a teljes minta 8,3 százalékát. Ha összehasonlítjuk az ábrákat, szembetűnő, hogy az első és a második nagyon hasonlít egymásra, a harmadik pedig nagyban különbözik ezektől.

Megvizsgáljuk azt az esetet is, amikor az értékelőfüggvény legfontosabb komponensét, az anyagot kihagyjuk az értékelésből. A *4. ábrán* ezt láthatjuk a nyerő és a vesztő fél szempontjából, az *5. ábrán* pedig a végső pontértéket — a nyerő és a vesztő fél függvényértékének különbségét.

Ha megvizsgáljuk az értékelőfüggvény-komponensek átlagának maximumát külön-külön, akkor a következő értékeket kapjuk:

Komponens	Maximum
Pszeudolegális lépések	41
Különböző megtámadott mezők száma	38
Centrumkontroll	184
Levy mozgékonysági függvénye	37
Bástyabónusz	29
Vezérbónusz	18
Futóbónusz	18
Megtámadott saját mezők száma	50
Megtámadott ellenséges mezők száma	17

Abban az esetben, ha olyan értékelőfüggvényt alkotunk, ahol mindegyik komponens pontértéke nulla és száz közé esik, a következő súlyozó kvócienseket kell használni az intervallum növelése vagy csökkentése érdekében:

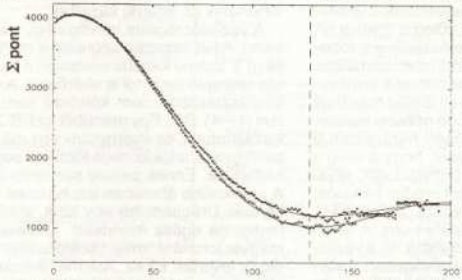
Komponens	Kvóciens
Pszeudolegális lépések (P)	2,4
Különböző megtámadott mezők száma (D)	2,6
Centrumkontroll (C)	0,54
Levy mozgékonysági függvénye (L)	2,7
Bástyabónusz (R)	3,4
Vezérbónusz (Q)	5,6
Futóbónusz (B)	5,6
Megtámadott saját mezők száma (OW)	2,0
Megtámadott ellenséges mezők száma (OP)	5,9

Ezeket a faktorokat felhasználva a következő függvényt kapjuk:

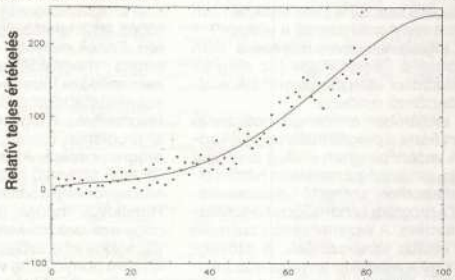
$F_{ú}$=1000•M+2,4•P+2,6•D+0,54•C+2,7•L+3,4•R+5,6•Q+5,6•B+2,0•OW+5,9•OP

Az ilyen formán módosított új értékelőfüggvényt a *6. ábrán* láthatjuk.

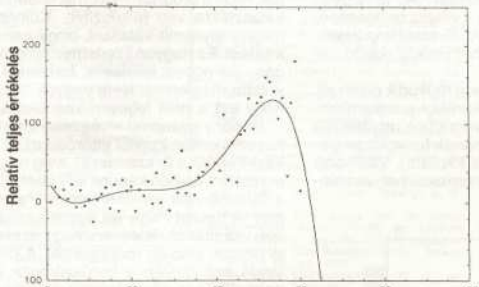
Kovács P. Attila



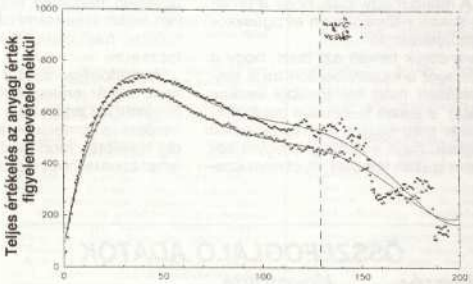
1. ábra Féllépésszám



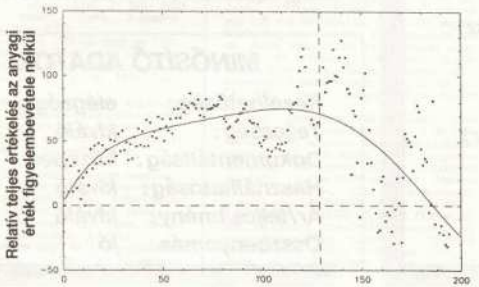
2. ábra Féllépésszám



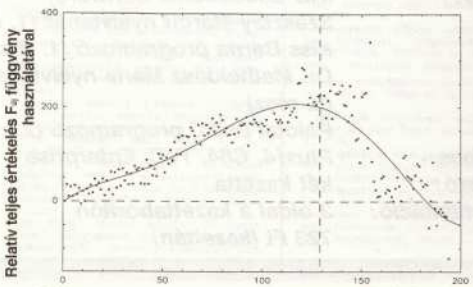
3. ábra Féllépésszám



4. ábra Féllépésszám



5. ábra Féllépésszám



6. ábra Féllépésszám

Egy újabb angolnyelv-oktató program

„Két lépés előre — egy lépés hátra". — Ez a klasszikus cím tolakodott a gondolataimba, amikor összegeztem benyomásaimat a Novotrade terjesztésében forgalmazott Opening Practice IHB programcsomagról. Régi igazság, hogy a jóban is lehet hibát találni, és hogy hibaok lehet az is, ha valami nincs benne egy termékben.

Az „újabb" jelző arra utal, hogy a cikksorozatban már volt szó egy angolnyelv-oktató programról (Nyelvvizsgateszt, szintén az IHB-től). Azt nem győztem dicsérni. Egyszerű, de mégis sokoldalú vezérlőprogramja meglehetősen nagy adatbázison dolgozott. A most elemzett program a mellékelt leírás szerint a nyelviskola oktatóprogramjainak eddig leszűrt tapasztalatait egyesíti a programcsomag vezérlőprogramjában. Ezzel van a fő bajom. Habár az IHB programozói tényleg ügyesek, nem értem, hogy a dokumentációban ígért tapasztalatok közé miért nem fértek be a programozott oktatásban kialakított hagyományos vezérlő algoritmusok (Skinner, Crowder stb.). Nem hiszem, hogy azokat már idejét múltnak kellene tekinteni. Kellemetlen, de felteszem a kérdést: lehet, hogy az IHB szerzői is csak írnak és nem olvasnak, mint annyi más szerző a világon?

A tanuló teljesítményének mérésére nem igazán alkalmas a Secret Page (az elég jól megoldott feladatsor után megjelenő titkos oldal) jellegű ösztönző rendszer.

Még egy, elöljáróban említendő, bosszantó dolgot vettem észre a programcsomaggal kapcsolatban. A vezérlőprogram elvileg önmagát dokumentálja, tehát még a mellékelt három oldalnyi, kazettaborítón szereplő „dokumentáció" sem kell a program biztonságos használatához. Pont fordítva. A kazettaborítón szereplő információk inkább félrevezetőek. A szövegben az áll, hogy a modulok anyaga majdnem tökéletesen illeszkedik az eredeti nyelvleckékhez, csak itt-ott vettek fel néhány új szót a téma bővítésére. A mondat úgy igaz, hogy a programcsomag sokszor tökéletesen elrugaszkodik az eredeti tananyagtól.

Miközben a derék tanuló azt hiszi, hogy a programcsomagot a kazettaborítón túl a tankönyvek alakjában még két további vaskos „dokumentáció" egészíti ki, elhűlve tapasztalja, hogy eleinte még csak a szereplők nevei nem stimmelnek, majd a szöveg is egyre kevésbé. Ez nem igazán fair play. Ilyet nem szabad csinálni a vásárlóval! Nem kellett volna lespórolni még egy ahhoz hasonló segédkönyvecskét, mint amit a hanganyagoknál használnak. Egyes frázisok fordítására szerintem rá sem jön a tanuló, és mivel a program a helyes választ semmi szín alatt el nem árulja, sohasem tudhatja meg. Azok örökre a program titkai maradnak, kivéve, ha valaki feltöri a programvédelmet, mert azután már kinyomozhatók.

Programvizsgálatunk tárgya tehát ezúttal a Novotrade és az International House Budapest (IHB) Nyelviskola Opening Practice programcsomagjának Plus/4-es változata, melynek összefoglaló adatait a szokásos táblázat mutatja.

A programcsomag óvatosan bánik a számítógép adta lehetőségekkel, főleg a grafika terén. Ennek mélyebb oka valószínűleg a többgépes megvalósítás kompromisszumaiban keresendő (a Plus/4-es változatban a színösszeállítások több szituációban is elég rossznak bizonyultak). A legidegesítőbb néhány kezelési probléma. Emiatt különösen haragszom a programozókra. Az a módszer, hogy a Help a SPACE billentyű hatására aktivizálódik, teljesen agyonvágta a billentyűzet eredeti funkcióit. Rendkívüli módon akadályozza a haladást, hogy szavanként kell RETURN-t ütni. A SPACE sokkal jobb szólezáró lett volna, és a vezérlőt nem bonyolította volna el túlságosan. Talán más gépek botkormány-kezelése indokolta ezt a meglehetősen rossz húzást? Az is nagyon ügyetlen megoldás, hogy a válasz befejezéséhez előbb át kell térni SPACE-szel Help üzemmódba, mert csak ott lehet Ready (kész) választ adni.

A random-generátor sem működik olyan jól, mint a már emlegetett korábbi programban. Sorozatban adják fel a programok ugyanazt a kérdést (a Jumbled — keverék funkcióban pedig tökéletes mondatot is kaptam). Vitatkozni lehet azon is, hogy a program csak helyes shiftben fogadja el a betűket. Erre a használati utasítás nem utal, így egyes alkalmazók, különösen otthoni használatban, reménytelenül elakadhatnak egy-egy ponton, mert képtelenek rájönni a hiba okára. (Öreg számítógépes rókáknak persze ez nem kunszt, de másokra is gondolni kell!)

A két kazettán két különböző stílusú anyag található, bár a vezérlőprogramok alapelvei hasonlóak. Az egyes kazetták tartalma: Mini párbeszédek (1. számú kazetta); Szövegek — történetek (2. számú kazetta).

A vezérlőprogram mindig négy menüpontot kezel. A két kazettán azonban a menük eltérőek (a 2. számú kazetta esetében a kazettaborítón szereplő leírástól is eltérőek!). Az almenük kiválasztásában nem kötelező sorrendet tartani (1—4). Sőt. Egy menüből az ESC gombbal kiszállhatunk, és áttérhetünk egy másik menübe. A gyakorlatok közben többféle segítséget is kérhetünk. Ennek persze pontlevonás az ára. A legolcsóbb általában az, ha csak egy betűt kérünk. Drágább, ha egy szót, a legdrágább pedig, ha egész mondatot. Kérhetjük még a magyar fordítást, amiért szerencsére nem mindig jár levonás. Ez az „Ask me" (kérdezz!) funkcióban magától értetődő, hiszen jobb ha tudjuk, hogy a program milyen lehetséges kérdésválasztékra van felkészítve. Különben hümmögve olyasmit válaszol, hogy nem érti jól a kérdést. Ez nagyon szellemes programozói fogás, de egyes kérdések fordítását képtelen voltam megfejteni. Nem vagyok benne biztos, hogy ezt a drillt teljesen kezdőknek szánták!

Néhány gyakorlat elvégzése után (megint a kazettaborítón írtaktól eltérően ez az 1. számú kazettán 10, a 2. számún 5), elég nagy szerzett pontszám esetén kapunk egy játéklehetőséget a fentebb már említett Secret Page formájában. A Secret Page-en egyes szavak intenzíven világítanak, jelezvén, hogy ezeket az elért jó pontok alapján megnyertük. A többi kitalálandó szó helyén a betűszámnak megfelelő pont áll (a normál feladatok szavai esetén ugyanez a helyzet). A szavak megadásakor a nagybetűket is pontosan kell írni. Míg a normál gyakorlatokon rontás esetén csak pontlevonás

ÖSSZEFOGLALÓ ADATOK

Forgalmazó:	*Novotrade*
Terméknév:	*Opening Practice*
Szerző:	*IHB Educational Software* *Szesztay Margit nyelvtanár (1. rész)* *Kiss Barna programozó (1. rész)* *Dr. Metheidesz Mária nyelvtanár (2. rész)* *Palotai Enikő programozó (2. rész)*
Géptípus:	*Plus/4, C64, TVC, Enterprise*
Hordozó:	*két kazetta*
Dokumentáció:	*3 oldal a kazettaborítón*
Ár:	*723 Ft (kazettán)*

MINŐSÍTŐ ADATOK

Kezelhetőség:	*elégséges*
Teljesség:	*kiváló*
Dokumentáltság:	*közepes*
Használhatóság:	*kiváló*
Ár/teljesítmény:	*kiváló*
Összbenyomás:	*jó*

van, addig a Secret Page-en egy rontást még megbocsát a program, és a Help ingyen kérhető, bár olykor nem is sokra megyünk vele.

A 2. számú kazettára érvényes leginkább, hogy messze elrugaszkodik az eredeti tananyagtól. A feladatok azonos modulban is jelentősen nehezebbek az 1. számú kazettán található feladatoknál. Ezekre tehát csak a tankönyv és az 1. számú kazettán található modul alapos feldolgozása után ajánlatos áttérni.

A programcsomag felöleli a Opening Strategies tankönyveknek mind a 15 gyakorlatát, és mint arról már szó volt, azon túlmutató anyagot is ad. Végeredményben jó vezérfonalként használható, így a tanár szinte tökéletes pótlására is alkalmas — kivéve a kiejtés gyakoroltatását, amit a hanganyag pótolhat. A tanulót rákényszeríti, hogy előbb foglalkozzon a könyvbeli tananyaggal és hanganyaggal, csak azután kezdjen hozzá a számítógépes drillhez. Aki vét ez ellen, az előbb-utóbb arra kényszerül, hogy visszalépjen a tananyag valamelyik korábbi fázisába.

A programcsomag ára igen kedvező, ha azt vesszük, hogy tanári vezényletet tud pótolni. Vegyük tekintetbe, hogy a könyvek 15 leckéjével három szemesztert is el-elbajlódna egy tanfolyamon az ember, ami nem kis pénzbe kerül manapság.

A programcsomag végül is „jó"-nak minősíthető. Ennek az érdemjegynek a részleteit táblázatba foglaltuk.

A programcsomag ajánlható mind tanfolyami, mind egyéni tanulás segítésére. Minden — megfelelő számítógéppel rendelkező — művelődési házban ott volna a helye. Sőt azt is lehetne javasolni, hogy a közkönyvtárakban a nyelvkönyv mellé a programkazettát is vegyék meg, és azt is kölcsönözhessék az olvasók.

Zsadányi Pál

ADOK—VESZEK—CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: kereskedelmi tevékenységet folytatóknak gépelt soronként (60 karakter) 100,- Ft, másoknak az első sor 50,- Ft, minden további sor 20,- Ft. Kérjük, hogy a hirdetés díját az NJSZT OTP V.ker. fiókjánál (218-98055) vezetett 508-8609 rendezvényszámlára vagy 1054 Bp., Báthori u. 16. címére fizessék be, rózsaszínű postautalványon (jelölve, hogy apróhirdetés), a befizetést igazoló szelvényt pedig csatolják a hirdetéshez. Hirdetéseiket a szerkesztőség címére várjuk (1371.Bp., Pf.: 433). Az NJSZT tagjai továbbra is kedvezményesen hirdethetnek (az első sor ingyenes), de kérjük, hogy adják meg tagsági számukat. Azokat a hirdetéseket, amelyek a hónap első napjáig megérkeznek, már a hónap végén viszontláthatják lapunkban.

ADOK

Amiga programok nagy választékban eladók! Érdeklődni lehet: Keresztes Gábor, Budapest, Laky köz 11. 1142. Tel.: 164-3452

AMIGA !!! THE WINNER IN HUNGARY !!! Legújabb programok minden mennyiségben. Szőnyi Laci, Budapest, Tavirózsa u. 5. 1161. Tel.: 184-8471

Amigához monitor, tárbővítő külső meghajtó eladó. Pető József, Szeged, Szilléri sgt. 49/A. 6723

Amiga programok lemezenként 30 Ft-ért, **C16, C Plus/4** programok 5 Ft-ért eladók. Tel.: 134-9278

Olcsón eladó 2 db **APPLE II+** típusú számítógép (6502 + Z80 processzor, Apple DOS + CP/M operációs rendszer, printer kártya, 80 karakteres kártya, 128 k RAM kártya, RS 232, IEEE 488, Óra stb.); 2 db dual MF 1800/900 típusú drive; 2 db NEC FD 1054 típusú drive. Érdeklődni: 8-16 óra között Kovács vagy Mayer, 166-5011/14-51 vagy 14-57 mellék.

Atari 800XL (64 k).gyári sztereo magnóval, joystickkal, játékprogramokkal (+ másolóprogram), szakirodalommal sürgősen eladó. (Irányár: 15 000 Ft). Pálmai Ferenc, Keszthely, Stromfeld A. u. 19. 8360

Atari 800XL-re játékprogramok, valamint TURBO-BASIC leírás olcsón eladó. Csere is érdekel. Tel.: 147-9484

C Plus/4 programokat adok és cserélek kazettán. A leveleket listával várom. Balázs Gábor, Budapest, Dinda u. 36. 1155

C Plus/4-es programok olcsón (8 Ft/db) eladók. Válaszborítékért listát küldök. Pápes Zoltán, Maglód, Ady E. u. 14. 2234

C64-es lemezek eladók (200 db) 100 Ft/db, szuper programokkal. Kérésre listát küldök. 1351-es egér, Citizen 120 D + interfész, magnó (6000, 25 000, 2000 Ft) eladók. Németi Ferenc, Budapest, Nagyenyed u. 8/a. 1182

C64-hez eladó új 1351-es egér és 1764-es RAM-bővítő (256 k-s) modul + tápegység. Árajánlatot kérek. Mészáros Tamás, Miskolc, Kondor B. u. 18. 3535. Tel.: 46/76-968

C64/II reset gombbal, magnó, cartridge turbókkal, 3x8 bites I/O port, kb 800 program 15 kazettán és 20 lemezen eladó. Irányár: 25 000 Ft. Érdeklődni lehet: Sajtos János, Székesfehérvár, Pf.: 152/M. 8001

Eladom új kivitelű **C64**-esem magnóval, joystickkal, joyballal, könyvekkel és 100 db programkazettával; külön-külön is!! Az árajánlatot tartalmazó levelet a következő címre várom: Egri Imre, Békésszentandrás, Pacsirta u. 6. 5561

Eladom **C64**-es régi és új programokkal teli kazettáimat, 65 db 5-től 60 percesig. Egyben szuper olcsón. Listát csak a küldött adathordozóra másolom rá. Oláh Lajos, Hort, Kossuth L. u. 147. 3014

C64-es programokat másolok! A programok ára 4 és 7 Ft/db. 700 db programról listát küldök! Bohács Tibor, Nagykálló, Petőfi u. 8. 4320. Te.: 346

C64-es szuper programok nagy választékban eladók kazettán és lemezen egyaránt garanciával, kedvezményekkel + 1 ajándék! Kérjen díjtalan tájékoztatót! Kiss Gábor, Győr, Venyige út 31. 9028

C64-re 1988/89-es programokat adok (20-25 Ft/db) és cserélek lemezen és kazettán. Levelezőpartnereket keresek! Molnár Tibor, Nyíregyháza, Kossuth u. 59. 4400

C64, VC-1541 lemezmeghajtó, MPS-801 nyomtató, datasett, 2 joystick, szakkönyvek, 130 db lemez programokkal, kiegészítőkkel eladó. Árban megegyezünk! Klazka István, Budapest XIII., Rozsnyai u. 5. II/46.

C64 programok olcsón eladók. Sok '88-'89-es programom van. C64 alapgép + magnó eladó. Micsik Gábor, Szeged, Szent Miklós u. 3. 6721. Tel.: 62/27-771

C64-es programokat adok kazettára (10 Ft/db). Kőhegyi László, Siklós, Dózsa u. 21. 7800

C64-re programokat adok kazettán és lemezen. Horváth Tibor, Győr, Tas vezér u. 35/A. 9012

+++ COMMODORE 64-esek figyelem!!! +++ **C64**-nek és tulajdonosuknak a legjobbat!! Eladók a legjobb programok lemezen! Csak 19 Ft/db. Ugyanitt 2 BIT D/A átalakító eladó!! Mágneslemezek üresen, originálban!! Válogatott prg-kazetták!! Cím: Kótai Balázs, Sopron, Margitbányai u. 37/a. 9400
Üdvözlet: CEMENT, FBI! FORM X-TEAM CODED BY ME!

C64-re több mint 1500-féle program eladó; 20 Ft/db. Kérésre listát küldök. Ugyanitt eladó egy adatmagnó C64-re, egy UNIVERSUM kétkazettás rádiómagnó és egy műanyag csillagküllős BMX kerékpár. Kasza Viktor, Siófok, Fenyves sor 11. 8600

C64-es programokat adok kazettán (8 Ft/db)! Válaszborítékot kérek! Holczmann Árpád, Győr, Felszabadulás u. 63. 9023

C64-es színvonalas programok lemezre eladók. 5 program felett 10% kedvezmény. Kérjen ingyenes tájékoztatót! Pintér József, Győr, Marx út 47. 9012

C64 + magnó + joystick eladó. Levélcím: Fogd Tamás, Ecser, Apponyi u. 8. 2233

Commodore 128D plusz 100 db lemez programokkal eladó. (C128 és 1571-es floppy egyben). Egyéves, új. Ár megegyezés szerint. Cím: Budapest XXII., Tenkes u. 7/A. 4 em. 15. 1225

Commodore 128D számítógép (VC 1571-es floppyval), lemezekkel, szakkönyvekkel, 2 db joystickkal 38 000 Ft-ért sürgősen eladó. Érdeklődni lehet telefonon: 56/20-495 (Debrecen).

Commodore 128D gépem eladom. Három elsődleges üzemmód: C128, C64, CP/M, beépítve 1571-es floppy. Plusz tartozék egy 128 k-s RAM-bővítő, BASIC-VOX magyar beszélőegység C64-re, 2 db joystick, 90 db diszk több kitűnő felhasználói programmal és leírással, játékokkal. Szinte teljes könyvdokumentáció (összesen 30) a különböző üzemmódjaihoz. Irányár: 60 000 Ft. Födi József, Szeged, Kállay Albert út 6-10. 6720

Eladó: **Commodore 640** rendszer – C128 alapgép, 640 k memóriabővítő, 2 db 1571 drive, 1902A monitor, Commander kazettás egység, 300 baud modem, COVOX hangdigitalizáló, SWL rádió dekóder, Quickshot joystick, teljes CP/M rendszer, Command center dobozban, hűtő berendezéssel. Eladási ár: 129 000 Ft. Tel.: 1556-643

Commodore 1581-es 3,5"-es floppy eladó. 800 kbájt formattált kapacitás, alkönyvtárszervezési lehetőség. Ugyanitt 360 kbájtos floppy-kontroller kártya + kábel eladó. Érdeklődni a miskolci (46) 35-238-as telefonon.

Commodore 1802-es színes monitoromat, kedvező áron, eladásra ajánlom. Érdeklődni: 15-től 20-ig (cím: Sopron, Paur Iván út 10.).

Enterprise programok eladók. 10-50 Ft. Kb. 1000 program! Válaszborítékért listát küldök. Érdeklődés kizárólag levélben! Zemen László, Budapest, Kada u. 141. 1104

Enterprise programok nagy választékban, lemezen is. C64 programcsere, eladás kazettán. VC 1541 floppyt vennék. Racskó Zsolt, Nyíregyháza, Hámán Kató u. 1. 4400

Sinclair ZX-Spectrum Plus + Kempston joy. + joy. interfész + Citizen 120D + printer-interfész + magnó + kb. 300 program + bőséges irodalom kedvező áron eladó! Cím: Bátorfi Zoltán, Rákóczifalva, Dózsa Gy. u. 17/a. 5085

Spectrum programok olcsón eladók. Enterprise programokat adok, veszek, cserélek. Cím: Sváb László, Budapest X., Pára u. 8. 1108

Spectrum-tulajdonosok! Új gazdát keres egy Seikosha GP-500AS printer egy interfész társaságában. Irányár: 15 000 + 5000 Ft. Érdeklődni lehet Ádám Gábortól Sajószentpéter, Kurucz utca 2. 3770 címen vagy munkaidőben a 48/11-211/2415 telefonon.

Spectrum 48/128 programok eladók! Mindig a legújabb, garantált minőségű szoftverek! Kompletteket 1 nap, külön válogatottakat maximum 3 nap alatt szállítom. Katalógusért küldjön válaszborítékot! Frim András, Pécs, Zrínyi u. 1. 7621

ZX-Spectrum (48 k), vadonatúj fólia, DK'TRONICS dupla interfész, mikrokapcsolós joystick, magnó, 53-55 kazetta - kb. 700 válogatott program, Spectrum Világok, könyvek, feljegyzések együtt vagy külön eladók. Irányár: 18 000 Ft. Érdeklődni: Kajtár Zsolt, Bonyhád, Zalka Máté utca 19. 7150

ZX-Spectrum 48, interfész II., szakkönyvek, programkazetták együtt 15 000 Ft-ért eladók. Sony 5 cm képátmérőjű Watchman mini tévé 13 000 Ft-ért eladó vagy C64, illetve C128 gépre cserélhető, értékegyeztetéssel. Bak Attila, Budapest, Népszínház u. 40. II/4. 1081

Videoton TVC 64 k + magnó + 600 prg. + könyvek olcsón eladók. Tel.: 1491-611

Akarsz csúcsszintű DEMO-t írni? Ennél mi sem egyszerűbb, ha POWER PACK-ot használsz!! Raszertrükkök, Sprite-animáció, soundmonitorok, packerek stb. Mindez EGY LEMEZEN a POWER PACK-on. Megrendelhető: Kótai Balázs, Sopron, Margitbányai u. 37/a. 9400. Üdv CEMENT, FBI, SYNDICATE!!!

ASSY V2.0 C64 mikroassembler + beépített editor + leírás + példaprogramok lemezen, utánvéttel 290 Ft. Káldi Péter, Budapest, Rezső u. 23. 1171

Áron alul, sürgősen eladó, garanciális Philips monochrom (zöld) computer-monitor 80. Cím: Szűcs Ilona, Hódmezővásárhely, Anna u. 2. 6800

Fordításokat készítek a **64'er Magazinból**. Több mint 3000 oldalnyi kész anyag! Kedvező árak. Textomat+, GEOS 1.2, Hi-Eddi, Giga CAD, Disc Wizard, Disc Demon stb. dokumentációk. Szolnoki Béla, Budapest, Pf.: 400. 1446. Válaszborítékot kérünk!

Joystick Szerviz a Flórián Áruházhoz közel! Javítás, magnófej beállítás. C64 játékprogramok kazettán és floppyn 10 Ft/db. Budapest III., Kerék u. 36. IV/24. Hétfőn és szerdán 17-től 19-ig.

Jutányos feltételekkel vállalalok középfokú német nyelvvizsgára való felkészítést. C64 vagy C128 és floppy szükséges. Érdeklődni: Szolnoki Béla, Budapest, Pf.: 400. 1446

NA VÉGRE! Minden együtt. 20-szoros turbo, programtörő és RESET gomb, szövegszerkesztő, monitor, nyomtató-interfész, óra, számológép és egyéb nyalánkságok, és mindez EGY cartridge-ben C64 és C128-hoz! Kívánságra bővebb tájékoztatót küldünk vagy bemutatjuk használatát. Tel.: 1848-845

Olcsó HW-cikkek Commodore-hoz, Enterprise-hoz. Válaszborítékért prospektust küldünk. COMPUTEAM GM, Kaposvár, Berzsenyi u. 32. 7400

'RPR-210 bármely géphez illeszthető (C64-hez is) Epson-kompatibilis nyomtató, Enterprise-hoz diszkvezérlő, többféle géphez illeszthető (PC, EP-128, TVC, Homelab) Eprom-égető (2716-27256) olcsón eladó. Mészáros Gyula, Budapest, Zsíroshegyi út 110. 1029. Tel.: 1699-499/1687, 10-től 16 óráig.

Új Vadnyugat - magyar nyelvű kalandjáték C64-re (Méltó követője A bosszú c. programnak!) Ára lemezen 370, kazettán 340 Ft. A hordozó árával együtt! Megrendelhető: Rátkai István, Esztergom, Bocskoroskúti út 28. 2500. (L.: TOP-LISTA)

3M-es, DS, HD floppy lemezek eladók. Tel.: 1860-609

CSERÉLEK

Általános iskola BASIC-szakköre **C16, C Plus/4, C64-re** készült, BASIC- és szaktárgyi oktató, ill. játékprogramokat cserélne lemezen vagy kazettán, iskolai csoportokkal. Listát kérek! Általános, Iskola, Nyirád, Szabadság u. 1. 8454, Hoffmann István.

C16 + 1531 +1541 II. + Citizen 120 D-t IBM PC-re cserélnék! Lilli József, Csolnok, Temető út 29. 2521

C Plus/4 játék-, felhasználói- és oktatóprogramokat cserélek, elsősorban lemezen. Kálmán Albert, Eger, Rákóczi út 31. III/11. 3300. Tel.: 36/25-648

C Plus/4 programcsere! Színvonalas programok KAZETTÁN! Listát kérek és kérésre küldök! Levelére egy héten belül válaszolok! Kresák Gergely, Pásztó, Arany János u. 5. 3060

C16, C Plus/4 és C64-es programjaimat kínálom fel cserére, szalagon és lemezen! Főleg oktató- és játékprogramok érdekelnek, lehetőleg listát kérek! Várom a régi cserepartnereim jelentkezését is! Hika György, Szeged, Sárosi u. 1/A. 6724. Tel.: 62/30-496 (17 óra után!). Csak cserélek!

C64 játék- és felhasználói programokat cserélek lemezen és kazettán. Oláh Zsolt, Miskolc, Laborfalvi u. 2. 8/4. 3525

C64-re programokat cserélek lemezen. Listát kérek és küldök, minden levélre válaszolok. Berendi Pál, Sásd, Fáy u. 30. 7370

C64-es programokat cserélek kazettán. Listát kérek! Kun István, Szolnok, Jászi Ferenc út 16. II/12. 5000

Enterprise programcsere lemezen is. Listát kérek! Cserealapom: kb. 550 program. Kusznyik Gábor, Debrecen, Virág u. 4. 4029

Enterprise 128-as számítógépemet 360 k-s floppyval, programokkal (dBASE-II, TURBO PASCAL, WORDSTAR is van) irodalommal, monitorral sürgősen elcserélném C64-es számítógépre. Érdeklődni telefonon: Burai, Debrecen 52/19-870 - 17 óra után.

Spectrum programokat cserélek! Listát kérek! Ábrahám Zsolt, Budapest, Újház u. 3. 1104

ZX-Spectrum 80 k számítógép programokkal eladó. Irányár: 9000 Ft. Kovács János, Alsóörs, Csokonai út 14. 8226

ZX-Spectrum 48 k-s programokat cserélek. Több száz programmal rendelkezem. Különösen a stratégiai és az akciójátékok érdekelnek. Listát kérek! Illés Tamás, Nyíregyháza, Fészek u. 200. 4400

Ez a rovatunk **KODEX 2000** szövegszerkesztő géppel készült.

Pongor György:
PASCAL. Programozás és algoritmusok
(Budapest, 1988. Novotrade Rt., 440 oldal.
Ára: 239 Ft.)

A Pascal nyelv magas szintű programozási nyelv, ezért használatához nem is kell ismerni a számítógép működését. Niklaus Wirth, zürichi professzor a Pascal nyelvet oktatási célra tervezte. Olyan nyelvet akart definiálni, amely könnyen tanulható, jó programozási szokásokat sugall, illetve támogat, és kisméretű számítógépeken is jól megvalósítható. A nyelv első alakját 1968-ban fogalmazta meg, az első Pascal fordítóprogram 1970-ben készült el, a nyelv első, szélesebb körben is ismert leírása pedig 1972-ben jelent meg. A nyelv gyors elterjedése és a vele kapcsolatban összegyűlt tapasztalatok 1982-ben vezettek a nemzetközi szabvány megalkotásához. Ez sok kicsi és néhány lényeges eltérést tartalmaz az „eredeti", Wirth által leírt nyelvhez képest.

A szabványos Pascal nyelvet ismertető kiadvány magyarul még nem jelent meg. Az eddigi legújabb, a szabvánnyal foglalkozó magyar nyelvű könyv, a SZÁMALK kiadványa is csak a szabvány előzetes változatát vehette figyelembe.

A Novotrade kiadásában megjelent könyv a már véglegessé, nemzetközileg elfogadottá vált nyelvet tárgyalja, de nem egyszerűen a szabvány fordítása. Tartalmazza mindazt, amit a Pascal nyelv szabványa, de az anyag csoportosítása elsősorban didaktikai szempontok szerint történt. Egy-egy nyelvi elem tárgyalásakor egy helyen adja meg az összes olyan szabályt, amely az adott nyelvi elemre vonatkozik. Így sokkal könnyebb kézikönyvként is használni a kiadványt.

A bemutatott programok, illetve programrészletek úgy vannak megválasztva, hogy a tipikus programozási feladatokat, a nyelvi elemek tipikus alkalmazását mutassák be.

Bakos Tamás — Zsadányi Pál
Operációs rendszerek. 1—2. kötet
(Budapest, 1989. SZÁMALK.
Ára: 496 Ft.)

A szerzők művüket bevezető tankönyvnek szánják, ezért igyekeznek nem túl mélyre ásni az operációs rendszerek elméletébe, amelyről sok tekintetben ma is elmondható, hogy nem alakult ki elégé.

A téma tárgyalását a 80-as évek elején megjelent, új, második generációs operációs rendszer szakkönyvekben terjedő irányzatával indítják. Míg a korábbi könyvek egy-egy konkrét operációs rendszerhez kapcsolódtak, addig az új generáció megpróbálja általánosítva összefoglalni a szakma — különösen a mikrogépek forradalma kapcsán szétzilálódott — ismereteit.

A kötet tárgyalásmódja az „európai logikát" követi, amennyiben egy-egy téma bemutatásánál általában az elméleti alapokból indul ki, és csak később ad gyakorlati példákat.

Az operációs rendszerekről sokan úgy beszélnek, mint egy szükséges rosszról, ugyanakkor a mai alkalmazók zöme csupasz hardvert még sose látott, és aligha tudna vele mit kezdeni. A hardvert tökéletesen elfedi elölük a szoftver, amelynek legalapvetőbb része a többnyire operációs rendszerként emlegetett rendszerszoftver.

Az operációs rendszerek történetileg az akkor még méregdrága nagyszámítógépek jobb kihasználására születtek. Fő elvük ezért a hardver hatékony kihasználásának biztosítása volt. Közben az árak zuhanásával, majd a minik és mikrók megjelenésével a körülmények erősen megváltoztak. Ma a felhasználó jó kiszolgálása fontosabb szempont, mint a hardver hatékony üzemeltetése. Az operációs rendszerekre vonatkozó erős kritikai megjegyzések alapja éppen az, hogy a nagygépes operációs rendszerek meglehetősen rugalmatlanul reagáltak erre a kihívásra, ami a minik és mikrók erős térnyeréséhez vezetett. A miniszámítógépes, de különösen a mikroszámítógépes korszak operációs rendszerei sokkal jobban idomultak a követelményekhez. Gyökeresen új szoftvermegoldásokat hoztak létre az erőforrásszegény környezetre, teljesen elrugaszkodva a nagygépes tapasztalatoktól. Nem csoda, hogy a korai „első generációs" operációsrendszer-szakkönyvek inkább praktikus célokat követtek.

A szerzők célja, hogy megpróbálják rendszerezni a témakör legfontosabb alapelveit, majd példák kapcsán bemutassák az eltérő megvalósítási módokat. Eközben érintik a legújabb irányzatokat is (IBM OS/2, Unix-verziók, NOVELL stb.), de kerülik az iránymutató jóslatokat.

Czerwinski, M.:
Mikroszámítógépek üzemzavarainak elhárítása
(Budapest, 1989. Műszaki Könyvkiadó, 141 oldal. Ára: 135 Ft.)

A számítógép üzemzavarainak alapvetően két forrása lehet: a hardver és a szoftver.

A könyv első része a hardverhibákkal foglalkozik. Az ilyen hibák sokszor egészen egyszerűek: például egy kiégett biztosíték. Vannak azonban olyanok is, amelyeket ennél már kissé nehezebb megtalálni és elhárítani, például azokat a hibákat, amelyeket a villamos hálózaton keresztül érkező zavarok okoznak.

Különös figyelmet szentel a szerző a külső tárolóknak: a kazettáknak és a mágneslemezeknek. A könyvből megtudható minden, ami fontos ezek tárolásával, kezelésével, élettartamával, karbantartásával, tisztításával, valamint az adatbiztonsággal kapcsolatban. Foglalkozik a képernyő és a nyomtató lehetséges hibáival is. A kötet második része a szoftverhibákat veszi sorra. Már az első programozási kísérletek során kiderül, hogy egy BASIC-értelmező egészen makacs jószág lehet. A legkisebb elütést is lakonikus „SYNTAX ERROR" üzenettel bünteti. Ha azután egy valamivel nagyobb programot billentyűz be a felhasználó, akkor gyakran az angol nyelvű hibaüzenetek és vészjelzések egész áradata zúdul a fejére. Sok esetben a felhasználói kézikönyv sem segít.

A könyvben megtalálható több BASIC-változat valamennyi hibaüzenetének teljes jegyzéke. Minden hibához megadja a lehetséges okokat, és ezt példával is szemlélteti. Ezután részletes tájékoztató következik a hibák kijavításáról. Külön fejezet tárgyalja a programok tesztelését. Itt megtalálhatók azok az egyszerű módszerek, amelyekkel a programhibák nyomára bukkanhatunk. A szerző ismerteti, hogyan bánjunk a tesztelési segédeszközökkel, amelyeket már számos gépbe beépítettek, hogy ezzel is megkönnyítsék a programozó életét. Ha a technika ördöge tréfát űz az olvasóval, vegye csak le ezt a könyvet a polcról!

Balázs Judit — Kertes Klára:
Táblázatkezelő programok.
INTERCALC, CALC-GRAPHIC
(Budapest, 1988. Műszaki Könyvkiadó, 75 oldal.
Ára: 95 Ft.)

A pénzügyi világban, a gazdasági életben és a szinte mindenütt szükséges adminisztráció során nagyon sok hasonló, sőt egyforma feladatot kell elvégezni. Az üzleti életben mindennap tudni kell, hogy mennyi volt a bevétel és mennyi a kiadás. A bankban is mindig azt tartják nyilván, ki mennyi pénzt tett be és mennyit vett fel. Ezeknél sokkal összetettebb, bonyolultabb számítássorozatokat kell végezni például egy statikai számításnál vagy egy költségvetés tervezésekor.

A nagyszámítógépekkel ezek közül igen sok feladatot meg lehet oldani, de korábban az egyes feladatokra a szakemberek bonyolult, hosszú programokat írtak. A személyi számítógépek megjelenése nagy változást hozott ezen a területen is. A számítástechnikában élenjáró országokban kidolgozták azt a technikát (szoftvert), amely képessé teszi a számítógépet és alkalmazóját arra, hogy különösebb előképzettség nélkül is megértsék egymást. Az utóbbi évek egyik legismertebb ilyen szoftverterméke az USA-ban kifejlesztett VISICALC nevű program. Ez igen alkalmas a bevezetőben említett feladatok megoldására. Az eredeti VISICALC program megjelenése óta számos számítógéptípusra írtak hasonló programokat. Ezek közül például a CALC RESULT, a Lotus 1-2-3 vagy a Symphony személyi számítógépekre készült.

A könyv célja a VISICALC típusú programok bemutatása, CALC program néven. Legfőbb erénye az emberközeliség és az, hogy az alapszabályok rövid idő alatt elsajátíthatók.

Milyen legyen a molekula?

A vegyipari, kémiai számítástechnikai alkalmazásokra szakosodott CompuDrug Kisszövetkezet szakértői programjainak körét új taggal bővítette: a HazardExpert segítségével előre felmérhetők a különböző kémiai folyamatok. Így a különféle vegyészeti kutatások során igen sok idő és pénz takarítható meg. A szoftverrel nyert ismereteket a kutatók már a vegyület tervezésénél is felhasználhatják, irányítani tudják a különféle molekulák leendő tulajdonságait. A HazardExpert bevezetésével a kisszövetkezet arra számít, hogy programját külföldön is megvásárolják. Erre különösen azokban az országokban van remény, ahol nagy összegeket költenek környezetvédelemre.

Gigantikus!

A nagy kapacitású számítástechnikai rendszerek kiépítéséhez egyre nagyobb kapacitású és egyre kisebb méretű háttértárolókra van szükség. Ezt az igényt igyekszik kielégíteni az amerikai Imprimis Technology cég legújabb fejlesztése, az 5,25 inches, 1 Gbájt nem formattált kapacitással rendelkező Wren VII típusú merevlemez-meghajtó. Az egység író-olvasó fejének és a lemezeknek az előállításához gyakorlatilag a 760 Mbájtos elődnél alkalmazott technológiát vették alapul, nem növelték meg sem a felírási sűrűséget, sem a sávsűrűséget. Az újdonság a két évvel ezelőtt kifejlesztett bitzónás adatrögzítés használata, amellyel a meghajtó tárolókapacitását 35 százalékkal sikerült megnövelni a hagyományos adatrögzítési eljárásokhoz képest. A Wren VII átviteli sebessége 15—21 Mbit/s között van, átlagos fejpozícionálási ideje 16,5 ms, így a nagy tárolókapacitás gyors hozzáférési idővel párosul.

Győri kesztyű

A Glovita Kötöttkesztyűgyár győri új üzemének 2700 négyzetméteres csarnokában még a nyáron megkezdődött a próbaüzem. A csarnokban 360 elektronikus vezérlésű gép — közöttük jó néhány speciális berendezés — működik. A mintázatot is számítógéppel tervezik, s jellemző a tervezést támogató rendszerre, hogy nemcsak kesztyű, illetve zokni készítésére alkalmas. Az üzemben naponta 50—55 ezer pár bőrkesztyűhöz elegendő bélést kötnek, s évente mintegy 13 millió pár divat- és munkavédelmi kesztyű hagyja el a gyárat.

Beültetőrobot

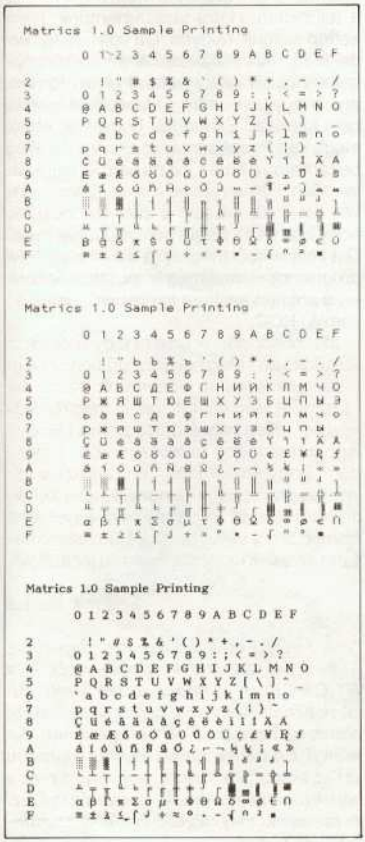
A Mikroelektronikai Vállalat a hazai felhasználók előtt elsősorban mint alkatrészgyártó ismert. Tevékenységében azonban jelentős szerep jut a berendezésfejlesztésnek és -gyártásnak is. A MEV az OMFB közvetítésével és anyagi támogatásával megvásárolta a felületszerelő robotok fejlesztése és gyártása terén sokéves tapasztalattal rendelkező Eurosoft Robotique francia cég berendezésének licencét. A számítógéppel vezérelt automatikus beültetőrobotból, amely a mikroalkatrészeknek a kártyán való elhelyezésére szolgál, az idén mintegy tíz darab készül exportra.

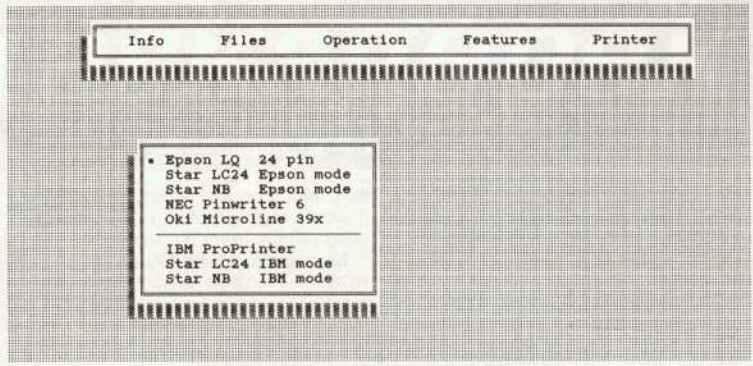
Karakterszerkesztő

Itthon is egyre jobban terjednek a 24 tűs mátrixnyomtatók, közülük is legfőképpen a Star Micronics Ltd. termékei, amelyek versenytársaik — többnyire Epson LQ nyomtatók — közül elsősorban a nagy teljesítményükhöz képest szokatlanul alacsony árukkal tűnnek ki. Bár a Star nyomtatókat manapság már teljes, CWI-szabványú magyar karakterkészlettel szállítják, ez nem oldja meg a más nyelvek, más betűformák, más karakterkészletek, speciális jelek használatát. Ezért dolgozta ki a Trimontána GMK a Matrics nevű karakterszerkesztő programját, amely könnyűvé, kényelmessé teszi az új karakterek tervezését. A menüvezérelt, egérrel is kezelhető, egyszerűen elsajátítható működésű és mindemellett rendkívül olcsó — egy-két ezer forintba kerülő — program olyan, önállóan használható, letölthető betűkészleteket állít elő, amelyeket egyéb programcsomagokkal, szövegszerkesztőkkel is használhatunk. A program használatához az IBM XT/AT-kompatibilis számítógép és a nyomtató kivételével semmi egyéb nem szükséges, még grafikus kártya sem.

A program jelenlegi, 1.0-ás verziója az Epson LQ sorozatával és az IBM Proprinter X24-es nyomtatóval kompatibilis készülékeket támogatja. A következő változat előreláthatólag kiegészül a 9 tűs nyomtatók támogatásával és további, a kényelmet fokozó funkciókkal is.

A Matrics segítségével tervezett magyar és cirill betűs betűkészlet vázlat- és levélminőségű formában





A Matrics menüképe

Egymorzsás Arcnet

A PC-hálózatoknál gyakran használják az Arcnet típusú hálózati kártyát, amelyet igen sok áramköri tokból építettek fel. Ez nemcsak elfoglalta a hosszú kártyák számára fenntartott PC-bővítőkártya helyét, de gyakran volt hibák forrása is, azon felül, hogy a szerelést és a bemérést megdrágította.

Az amerikai Standard Microsystems Corp. kifejlesztette az egymorzsás vezérlőt, a 90C23 Arcnet jelű áramköri elemet. Így viszonylag kevés többlet külső elem ráépítésével az igényeknek megfelelő csatolókártya tervezhető, amely már elfér a kevésbé használatos rövid kártyabővítő helyen is a PC-ben. A szuperáramkör kialakításánál az áramköri tokon belül több technológiát is kombináltak. Felhasználták a nagy integráltságú áramköri elemeket, amelyeket felületszerelve egyesítettek egyetlen hordozólapka felszínén. Az elemeket az egyes gyártók által felkínált szabványelemekből, azaz tokozatlan morzsákból választották a tervezők. Így az új Arcnet morzsa végső soron berendezésorientált áramkörökből készült el.

Alkatrész 3000 fiókban

Számítástechnikai alkatrészüzletet nyitott a győri Széchenyi István Főiskola. A boltban híradástechnikai és számítástechnikai eszközök alkatrészeit értékesítik. Az Elektromodul Kereskedelmi Vállalattal közösen létesített üzletet elsősorban a főiskola kiszolgálására nyitották meg, de nem hanyagolható el a közös nyereség sem: az idén már 14 millió forint értékű alkatrész értékesítését tervezik. Az üzlet egyik falán összefüggő, 3000 fiókból álló szekrényfal van, amelyben mintegy 4 millió forint értékű alkatrész várja a vásárlókat.

P.I.C.

A Panasonic Industrial Company elektronikus írógépei lesznek az elsők, amelyeket egy újfajta rövidítésprogrammal látnak el: a gépíróknak csak a szokásos vagy maga szerkesztette rövidítéseket kell bekopognia. A program a begépelt rövidítéseket automatikusan átszótározza, s a feloldott teljes szavas változatot írja ki. A FastType nevű program már eleve tartalmaz egy 1400 szavas szótárt, amely magában foglalja például a hét napjait, a megszólításokat és az üzleti világ szakkifejezéseit. A felhasználó ezenfelül beprogramozhat további rövidítéskódokat is, például saját cégének és termékeinek a nevét.

A rendelőben

A Hajdú-Bihar megyei Nagyhegyesen a körzeti rendelő egyik sarkában egy C64-es mikrogép segíti az orvos munkáját. A személyi és kórtörténeti nyilvántartás alapján a gép heti és napi munkaterveket készít, gyors, megbízható képet ad az orvos számára a megelőzéssel kapcsolatos teendőkről is. Egy jellemző példa az alkalmazásra: hogyan lehet egy cukorbeteg, magas vérnyomásos és gyomorpanaszoktól szenvedő páciensnél a szükséges vizsgálatokat úgy elvégezni, hogy a beteg a lehető legkevesebbszer utazzon be Hajdúszoboszlóra. A gép segítségével különböző demográfiai és betegséggyakorisági elemzések is készülnek.

Gyors tárolók

Míg a dinamikus memóriamorzsák gyártóinak többsége a tárkapacitás növelésére koncentrált, az NMB Technologies a termékeinek sebességére helyezte a hangsúlyt. Elsőként a 256 kbites áramkörök technológiáját gondolta át, és 60 ns-os eszközökkel rukkolt ki akkor, amikor a többiek még csak 80 és 120 ns-os morzsákat szállítottak. Gondos tervezéssel elérték, hogy ma már 1 Mbites áramköröket is készítenek 60 ns-os hozzáférési idővel, tűrhető kihozatal mellett. Az NMB tervezi, hogy beszáll a 4 Mbites piacra is. Koncepciója szerint egy új, nagy dielektromos állandójú anyagot hoz be a technológiába. A ferroelektromos réteg sokkal nagyobb töltést képes tárolni felületegységenként, mint a szilícium, így az NMB megmaradhat a jól bevált planár memóriacellák mellett, és a jövőben talán sokkal jobb kihozatallal fogja gyártani a 4 Mbites memóriamorzsáit, mint versenytársai.

Turbina-tomográf

Ezentúl a technikában is bevezetik azt a számítógépes tomográfiai eljárást, amelyet az orvosi diagnosztika már régen alkalmaz, s amelyleyel részletes keresztmetszetet lehet nyerni az emberi testről. Az amerikai General Electric kifejlesztett egy olyan számítógépes tomográfot, amelynek segítségével a repülőgépek turbináinak lapátjaiban ki lehet mutatni a finom repedéseket. A készülék nagyobb röntgensugár-energiával működik, és jóval finomabb képeket is állít elő, mint az orvosi diagnosztikai készülék. Segítségével már 0,2 milliméter nagyságú hibákat is láthatóvá lehet tenni. Az eljárást úgy automatizálják, hogy számítógép fogja végezni a képek kiértékelését és a hibahelyek felismerését is.

Az Olvasó írja

Hol vagy, Commodore Club?

Valamikor réges-régen a családfő úgy döntött, hogy igenis jó, ha az ember tagja egy klubnak, és kézenfogva — akkor még — kisfiát, vagyis engem, egy borús téli estén elindult megkeresni a klubot, azaz a HCC Commodore szekcióját. Több órát bolyongtunk a BME korántsem szívderítő épületei között — akkor állítólag még ott székelt a HCC —, de nem találtuk meg.

Teltek-múltak az évek, és eljött 1989 januárja, mikor is megjelent a Mikrovilágban a következő:

„Budapesti klubtükörkép

A HCC-hálózatba tartozó fővárosi klubok adatai:

Commodore: Budapest XI., Bocskai út 37. (TIT stúdió). Csütörtökön 18 órától várják az érdeklődőket. Felvilágosítást ad Simonyi Endre klubvezető a 154-090-es telefonszámon.

Ugyanitt lehet jelentkezni a most szerveződő 32 bitesek klubjához."

De ez előtt a rövid kis részlet előtt majd kétoldalas interjút olvashattunk Simonyi Endrével, akitől többek között azt is megtudhattuk, hogy: „. . . . az anyaszervezet, az amerikai HCC klub 1985-ben felosztott. A HCC nevet (amelynek H betűje nálunk homebrew-t, home-ot és Hungariant egyaránt jelent) jelenleg a magyar klubhálózat használja egyedül a világon."

Ennyi szép szó olvastán feltámadt bennem a kedv, hogy megnézzem ezt a klubot. Alig tudtam kivárni a csütörtököt. S mikor végre-valahára eljött a kitűzött nap, hat óra előtt tíz perccel már a TIT stúdió kapujában álltam. Első állomás a portás néni volt, aki érdeklődő kérdéseimre szívélyesen válaszolt. Tőle tudtam meg, hogy járnak ide számítógépes fiúk, menjek csak fel a kémiateremhez, már most is ülnek ott néhányan. Valóban ültek már ott néhányan. Mint később kiderült, mind akkor jöttek először. Vártunk. Közben feljött a portás néni, kinyitotta a kémiatermet. Bent egy darab számítógépet sem találtunk, de még a tartozékait se, egy árva színes tévén kívül. Nagyon elcsodálkoztunk. Ekkor megérkezett egy kis srác, aki elmondta, hogy ő már volt itt a múltkor. Akkor valaki más hozott gépet, és azt mondták neki, hogy most ő hozzon, de neki ezt otthon nem engedik meg. Sebaj! Pár percen belül beállított két másik srác egy Plus/4-gyel. Ekkor kiderült, hogy a színes tévé sem ide tartozik, mert nem sikerült beállítani a számítógéphez. A két fiú szintén először volt itt, csak biztos ami biztos, elhozták magukkal a saját gépüket is. Háromnegyed hétig vártam, akkor hazamentem.

Márciusban a Mikro '89-en megjelent a HCC is — meg én is. Egyből kiszúrtuk egymást! Megkérdeztem egy klubtagot — aki mellesleg egy Commodore 64 előtt ült —, hogy mikor, hol és hánykor tartják az összejöveteleket. Bármennyire hihetetlen, de azt mondta, hogy a TIT stúdióban, minden második csütörtökön este hatkor. Nem hittem a fülemnek. Már éppen tovább akartam állni. mikor valami belső hang azt súgta, hogy kérdezzem meg, mikor van a következő összejövetel. Ő szolgálatkészen felvilágosított, hogy ekkor és ekkor jöjjek, mert akkor lesz. Elmentem. Csodák csodája, a kémiaterem nyitva volt, elég sokan ültek bent, még felnőttek is voltak, gondoltam, most „elkaptam" őket.

Egy bökkenő azonban volt. Nem volt itt az a fiú, akivel beszéltem. Fél órán belül kiderült, hogy mindenki először van itt. A felnőtt bácsi pedig egy új közösségi ház vezetője volt, aki tanácsot kérni jött. Hazafelé a buszon ráébredtem a HCC működési elvére. Sok reklám, sok embert idecsal, megismerkednek egymással, otthonukban összejönnek, máris mehet a programcsere, meg van oldva a teremgond! Ezt az elképzelésemet alátámasztotta az is, hogy a két sikertelen próbálkozásom alatt több száccal vettem fel a kapcsolatot. Közülük eggyel még most is tartom!

Ezzel az újdonsült barátommal beszéltük meg, hogy találkozunk a következő csütörtökön ugyanitt. Ám ekkor felvilágosított minket a portás néni, hogy csak minden hónap második csütörtökén van klubfoglalkozás, hogy ez ki is van írva a táblára az utca sarkán. Valóban, de azért mentségemre szóljon, hogy a Mikrovilágban simán csak csütörtök szerepel, a fiú a Mikro '89-en is csak simán második csütörtököt mondott, s az általa említett időpontnak köze sem volt a hónap második csütörtökéhez.

Barátomat többé nem lehetett rábeszélni, hogy eljöjjön ide a város másik végéről. Azt azonban még elmondta, hogy ő is kint volt a Mikro '89-en, de nem akart HCC klubtag lenni, tekintve, hogy azt sem tudja, mi az. Ő csak sétált, s valaki megszólította, hogy nem akar-e csatlakozni a HCC-hez. Mikor megkérdezte, mit tisztelhet a HCC-ben, az illető meghívta, mondván, hogy jöjjön el, nézze meg, nem fog csalódni! Ugyanazt a dátumot kapta, amit én, így találkoztunk. Ennél nagyobbat nem tudott volna csalódni!

Május második csütörtökén elfelejtettem elmenni, így maradt június második csütörtöke. Minden nagyon jól indult. Hosszas unszolásra a messze lakó barátom is beleegyezett, hogy ad egy utolsó lehetőséget a HCC-nek. Odamentünk fél ötre. Kint volt egy tábla, hogy ma a kémiateremben a számítógép-építők (?) tartják összejövetelüket este hat órától. Azért mi gyanakodtunk, és megkérdeztük a portás nénit is. Azt mondta, igen, lesz klub, hatkor. Már éppen távozni készültünk, hogy addig elütjük valahol az időt, amikor a portás néni utánunk szólt: „Lehet, hogy csak ketten lesznek!" Nem mondom meg, mi járt ekkor a fejünkben. Hatra visszamentünk. Nem ketten voltunk, négyen. Ott volt már két érdeklődő, aki csatlakozni akart a klubmozgalomhoz. S ekkor valami olyan történt, amire egyikünk sem számított. A portás néni kifakadt. Elmondta, hogy neki ebből elege van. Ilyenkor már csak ő van itt. Rég bezárhatná a házat, ha mi nem jönnénk állandóan. Félreértés ne essék, ő nem minket hibáztat, hiszen ő mondja nekünk, hogy lesz HCC, dehát nem mondhat mást. Ő a központból megkapja a heti vagy havi programot — amit meg is mutatott nekünk —, s amiben valóban szerepelt: június 8., kémia: HCC.

Mit mondhattunk erre? Hogy nekünk is elegünk van? Meg hogy az idő pénz? Nem mondtunk semmit, elmentünk haza. Jut eszembe! Még mielőtt valaki megkérdezné, hogy miért nem érdeklődtem a Mikrovilágban megadott telefonszámon, elmondom, hogy érdeklődtem én, de azt a választ kaptam, hogy Simonyi Endre külföldön tartózkodik. A portás néni egyébként szintén tudni vélte, hogy a klubvezető nagyon elfoglalt ember, s gyakran jár külföldre.

Spányik Balázs

Nos, hm. És még egyszer hm. Sajnáljuk, hogy ez az ígéretes kezdeményezés — a HCC — ilyen szomorú sorsra jutott. Bízunk benne, hogy nem ez lesz az utolsó szó ebben az ügyben. Volt kollégáinknak, Simonyi Endrének megkíséreltük eljuttatni ezt az írást... Reméljük, hogy mire ezek a sorok napvilágot látnak, ismét teljes gőzzel és mindenki megelégedésére folyik a munka a HCC-ben.

Táskagépek

TOSHIBA T 3100

Egyre több, hálózattól független táskagép készül 32 bites Intel 80386-os processzorral. Az egyik újdonság a Zenith Turbosport 386. A 2 Mbájtos, 5 Mbájtra bővíthető operatív tár kényelmes feldolgozásszervezést tesz lehetővé. A 28 milliszekundumos hozzáférési idejű, 40 Mbájtos winchester tár a professzionális igényeket is kielégíti. A gép nikkel-kadmium akkumulátora két óra alatt a világ bármely részén, bármely konnektorról feltölthető. Az áramtakarékos CMOS morzsáknak, valamint a központi egység, a tárolóegység és a képernyő számára kialakított takarékautomata révén a Turbosport 386 akár négy órán keresztül is önállóan működhet. A felhasználó kényelmét szolgálja a levehető billentyűzet, és a hátulról megvilágított, nagyméretű és nagy felbontású folyadékkristályos képernyő, erős kontraszttal és 16 árnyalati fokozattal. A hazai piacon az elmúlt év novemberében jelent meg először forintért hordozható mikrogép a Controll kínálatában.

Sci-L

IBM PC/XT-KOMPATIBILIS PROFESSZIONÁLIS SZEMÉLYI SZÁMÍTÓGÉPEK

AJÁNLATUNKBAN ÚJ

P-16/PC alapkonfiguráció

— 8088-alapú központi egység (4,77/10 MHz)
— 640 kB RAM
— billentyűzet (84 gombos)
— HERCULES/színes csatoló
— monokróm monitor, 12"
— 360 kB floppy
— real-time óra
— soros-párhuzamos interfész
ÁR: 80 000,— FT+ÁFA

P-16/CX XT alapkonfiguráció

— 8088-alapú központi egység (4,77/10 MHz)
— 640 kB RAM
— billentyűzet (84 gombos)
— HERCULES/színes csatoló
— monokróm monitor, 12"
— 360 kB floppy
— ST 225 winchester+csatoló
— real-time óra
— soros-párhuzamos csatoló
ÁR: 105 000,— FT+ÁFA

Alapkonfiguráció-változatok

(Az alapkonfiguráció árához az alábbiakban feltüntetett összegek hozzáadandók!

1,2 MB-os floppyval	+ 5 000,— Ft+ÁFA
1,44 MB-os floppyval (3,5 inch)	+10 000,— Ft+ÁFA
Színes monitorral (CGA)	+12 000,— Ft+ÁFA
EGA monitorral és csatolókártyával	+46 000,— Ft+ÁFA
ST251 (40 MB, 40 ms) winchesterrel	+25 000,— Ft+ÁFA
ST251—1 (40 MB, 28 ms) winchesterrel	+35 000,— Ft+ÁFA
VGA monitorral és csatolóval	+75 000,— Ft+ÁFA
Operációs rendszerrel (MS—DOS 3.30)	+10 000,— Ft+ÁFA
ARCNET hálózati illesztőkártyával (2,5 Mbit/s)	+25 000,— Ft+ÁFA
12 havi garancia a konfiguráció árának +6%-a, 18 havi garancia a konfiguráció árának +9%-a.	
Üzembe helyezési díj: +3,8%.	

Garancia nélküli reklámáraink:

• TRS aszinkron terminál+kábel	9 000,— Ft+ÁFA
• TRS mátrixnyomtató+kábel (132 kar/sor)	14 500,— Ft+ÁFA
• FX—100 mátrixnyomtató+kábel (132 kar/sor)	49 900,— Ft+ÁFA
• RANK XEROX 4045 lézernyomtató	290 000,— Ft+ÁFA
• UNIBOARD billentyűzet (101 gombos)	4 900,— Ft+ÁFA

Bővítési lehetőségeinkről kérjék árjegyzékünket!

SZÁMÍTÁSTECHNIKAI INFORMATIKAI FEJLESZTŐ LEÁNYVÁLLALAT

1011 Budapest, Iskola u. 10.
Teletax: 35 39 15
Telex: 22 4599

Sci-L

Telefon: 1-154-065
1-350-180/180, 181, 182, 184